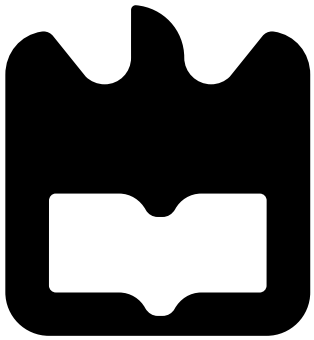
Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2013

Daniel Tavares Ferreira

Camada Protocolar de Aplicação para ITS-G5



Daniel Tavares Ferreira

Camada Protocolar de Aplicação para ITS-G5

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Doutor Arnaldo Silva Rodrigues de Oliveira, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Joaquim José de Castro Ferreira, Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor João Nuno Pimentel da Silva Matos
Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor José Carlos Meireles Monteiro Metrôlho
Professor Adjunto da Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco

Prof. Doutor Arnaldo Silva Rodrigues de Oliveira
Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (Orientador)

agradecimentos / acknowledgements

Em primeiro lugar quero agradecer ao meus pais que sempre me apoiaram nesta longa jornada académica. Também agradeço aos meus orientadores – Prof. Doutor Arnaldo Silva Rodrigues de Oliveira e Prof. Doutor Joaquim José de Castro Ferreira – pelo apoio prestado, bem como ao resto do pessoal da equipa do projeto HEADWAY do Instituto de Telecomunicações de Aveiro. Por fim, mas não menos importantes, quero agradecer a todos os colegas e amigos que me acompanharam na vida universitária.

Resumo

Os avanços nas comunicações sem fios permitiu que o conceito de comunicações veiculares (veículo-veículo e veículo-infraestrutura) surgisse como uma das principais soluções para reduzir a sinistralidade na estrada e aumentar o conforto das viagens. O interesse nesta área é crescente e levou a que estas comunicações fossem regulamentadas: nos EUA, pelo IEEE – conjunto de standards WAVE e 802.11p – e na Europa, pelo ETSI. Estas normas definem a função das várias camadas protocolares deste tipo de comunicações: desde a camada física até à de aplicação. São muitos os projetos em torno das comunicações veiculares e das suas normas, mas, ainda assim, existe muito a fazer nesta área.

Com este trabalho, pretende-se desenvolver *software* que implemente essas camadas protocolares superiores da norma Europeia EN ETSI 302 665 para comunicações veiculares. Esta implementação envolverá a recolha de informação de sensores para construção e envio de mensagens na rede veicular e, por outro lado, a sua receção e processamento, interagindo também com o utilizador. Assim, desenvolveu-se a parte da camada de suporte à aplicação responsável pela recolha de informação de sensores e troca de mensagens com a camada de rede. Além disso, desenvolveu-se a comunicação com um *smartphone* de forma a que o utilizador se possa ligar ao equipamento e dele receber e enviar alertas sobre os perigos mais próximos.

Esta dissertação resultou, então, num sistema que recolhe informação de vários sensores (do veículo, GPS e do *smartphone*), que gere as mensagens trocadas entre veículos de acordo com a norma Europeia do ETSI (TS ETSI. 102 637-2 v1.2.1 e TS ETSI. 102 637-3 v1.1.1, relacionadas com a norma EN ETSI 302 665) e que fornece a possibilidade de interface com o utilizador através de um *smartphone* Android. Devido à complexidade do *software* a este nível, optou-se por utilizar uma plataforma de *hardware* baseada num *Single Board Computer* a correr o SO Linux que dispõe de periféricos como recetor GPS e adaptador OBD-II (leituras dos sensores do veículo). O sistema implementado foi testado em ambiente de laboratório e cumpre, na sua maioria, as normas referidas em cima.

Abstract

The global advances in wireless communications drove the concept of vehicular communications (vehicle-vehicle and vehicle-infrastructure) to become one of the main solutions to reduce road hazards and to make road trip more comfortable. The attention towards this field of technology is rising and lead to the standardization about this type of telecommunications: In EUA, by IEEE – WAVE 802.11p – and in Europe by ETSI. These standards define the role of each protocol layer: from the physical layer up to the application layer. There are many projects around vehicular communications and around their standards, but still there is a lot to develop and accomplish in this area. With this work it is intended to develop software for this upper layers following the European standard about vehicular communications (EN ETSI 302 665). The development will include information gather from sensors with the objective of build and send messages on vehicular network, on the other way, it includes received message processing and also some kind of user interface. So, it was implemented the application support layer and a simple security application. The application support layer is responsible to gather information and exchange of messages in the network, it is also responsible for the communication with a smartphone that acts like a *Human Machine Interface* (HMI) from which the user sends and receives alerts about surrounding dangers.

In the end, this thesis result in a system that gathers information from various sensors (from vehicle, GPS and smartphone) and then, according to the European standards (TS ETSI. 102 637-2 v1.2.1 e TS ETSI. 102 637-3 v1.1.1, related to EN ETSI 302 665), builds and manage the exchange of messages between vehicles and offers the possibility of an HMI through an Android smartphone. With level of complexity the choice of a *Single Board Computer*(SBC) became natural. This SBC runs Linux and connects to various peripherals like GPS receiver and OBD-II reader (reads the sensors of the vehicle). Despite some limitations concerning the speed of data acquisition, this combination of hardware and software was tested on a laboratorial environment and is compliant, on it's vast majority, with the vehicular standards mentioned before.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

API *Application Programming Interface*

ASN.1 *Abstract Syntax Notation One*

BSM *Basic Safety Message*

BSA *Basic Set of Applications*

C2C-CC *Car-to-Car Communication Consortium*

CALM *Communications Access for Land Mobiles*

CAM *Cooperative Awareness Message*

CAN *Controller Area Network*

CCA *Cooperative Collision Avoidance*

CPU *Central Processing Unit*

DENM *Decentralized Environmental Notification Messages*

DER *Distinguished Encoding Rules*

DLL *Data Link Layer*

DRIVE-IN *Distributed Routing and Infotainment through VEhicular Inter-Networking*

DSRC *Dedicated Short-Range Communications*

ECDSA *Elliptic Curve Digital Signature Algorithm*

ECIES *Elliptic Curve Integrated Encryption Scheme*

ECU *Electronic Control Unit*

ETSI *European Telecommunications Standards Institute*

EWM *Emergency Warning Message*

FAST *Fix Adapted for STreaming*

GSM *Global System for Mobile Communications*

GPIO *General Purpose Input/Output*

GPS *Global Positioning System*

HMI *Human Machine Interface*

HEADWAY *Highway Environment ADvanced WArning sYstem*

I2C *Inter-Integrated Circuit*

ICSI *Intelligent Cooperative Sensing for Improved traffic efficiency*

IEEE *Institute of Electrical and Electronics Engineers*

ISA Interface de Suporte à Aplicação

ISO *International Organization for Standardization*

ITS *Intelligent transport systems*

IP *Internet Protocol*

IPv6 *Internet Protocol version 6*

LDM *Local Dynamic Map*

LLC *Logical Link Control*

LTE *Long Term Evolution*

MAC *Medium Access Control*

MIL *Malfunction Indicator Lamp*

MM-Wave *Millimeter Wave Networks*

MLME *MAC Layer Management Entity*

OBD *On-Board Diagnostics*

OBD-II *On-Board Diagnostics version II*

OBU *On-Board Unit*

OSI *Open Systems Interconnection*

PID *Parameter ID*

PHY *PHYsical Layer*

PLME *Physical Layer Management Entity*

POSIX *Portable Operating System Interface*

RAM *Random Access Memory*

RHW *Road Hazard Warning*

RSU *Road Side Unit*

SAE *Society of Automotive Engineers*

SBC *Single Board Computer*

SDK *Software Development Kit*

SO Sistema Operativo

SPI *Serial Peripheral Interface*

SQL *Structured Query Language*

SSL *Secure Sockets Layer*

TCP *Transmission Control Protocol*

UART *Universal Asynchronous Receiver/Transmitter*

UDP *User Datagram Protocol*

UMTS *Universal Mobile Telecommunication System*

UPER *Unaligned Packed Encoding Rules*

USB *Universal Serial Bus*

V2I *Vehicle to Infrastructure*

V2V *Vehicle to Vehicle*

V2X *Vehicle to Anything*

VANET *Vehicular Ad-hoc NETwork*

WAVE *Wireless Access in the Vehicular Environment*

WBSS *WAVE Basic Service Set*

WIMAX *Worldwide Interoperability for Microwave Access*

WLAN *Wireless Local Area Network*

WSMP WAVE *Short-Message Protocol*

Capítulo 1

Introdução

1.1 Enquadramento

As comunicações veiculares têm vindo a ser apontadas como uma das principais soluções para o problema da falta de segurança que o aumento do tráfego rodoviário provoca. Este conjunto de novas tecnologias permite que os veículos troquem informação entre si – sobre a sua rota, posição, velocidade, características e perigos detetados – para que, com base nesses dados, se possam aplicar mecanismos de prevenção de incidentes que põem em risco a vida das pessoas.

Reconhecendo a importância deste tipo de comunicações, em 1999, o IEEE definiu um conjunto de normas, inicialmente para uso nos Estados Unidos da América (EUA), sobre comunicações veiculares rádio de curta distância (*Dedicated Short-Range Communications* (DSRC)), designadas *Wireless Access in the Vehicular Environment* (WAVE). Mais tarde, normas com o mesmo cariz foram adaptadas e desenvolvidas na Europa (denominadas ITS-G5), primeiro pelo *Car-to-Car Communication Consortium* (C2C-CC) e depois regulamentadas pelo *European Telecommunications Standards Institute* (ETSI). Nestes standards está previsto que a troca de informação entre veículos seja feita através de mensagens, no caso americano, mensagens *Basic Safety Message* (BSM) e, no caso europeu, mensagens CAM e DENM. Estas serão transmitidas de forma segura/encriptada na banda dos 5.9GHz, mas, no caso europeu, está ainda previsto o uso de outras redes – por exemplo, celular – não só para precaver falhas na rede veicular, mas também para que possam ser oferecidos serviços de acesso geral à Internet.

Neste âmbito foram surgindo vários projetos, dos quais o *Intelligent Cooperative Sensing for Improved traffic efficiency* (ICSI) e o HEADWAY, a decorrer no Instituto de Telecomu-

nicações de Aveiro e onde esta dissertação se insere. Este projeto tem a finalidade de conceber e implementar uma plataforma modular e flexível que implementa as normas europeias e norte-americanas para comunicações veiculares na banda dos 5.9GHz (DSRC 5.9).

Atualmente, o trabalho desenvolvido sobre a plataforma HEADWAY ainda está muito focado no *hardware* e camadas protocolares inferiores, não oferecendo aplicações de segurança de alto nível ou interface com o utilizador. Assim, o trabalho desenvolvido ao longo desta dissertação pretende endereçar estes aspetos em aberto, isto é, acrescentando meios para o desenvolvimento de aplicações de segurança e interação com o utilizador.

1.2 Motivação

A Organização Mundial de Saúde prevê que, se não existir uma inversão, em 2020, os acidentes rodoviários serão a terceira maior causa de doença e lesão a nível global [1]. Além disso, segundo Meyer et al. [2], só nos EUA as despesas com acidentes e congestionamento rodoviário chegam aos 300 mil milhões de dólares por ano. Assim, torna-se imperativo desenvolver formas de combater os incidentes e melhorar a segurança nas viagens.

A indústria automóvel tem tentado, com algum sucesso, fazer frente a este problema através de sistemas avançados, como o Airbag e o ESP, que recorrem a vários sensores (acelerómetro, sensores de deformação, etc.) para poderem tomar decisões e melhorar a segurança do veículo e dos seus ocupantes. Estes sistemas e sensores já estão presentes na maior parte dos veículos recentes e, além disso, podem ser utilizados noutros sistemas de segurança uma vez que uma boa parte deles pode ser acedida diretamente através do interface normalizado OBD-II.

Por outro lado e como foi referido na secção anterior, têm vindo a emergir uma nova classe de redes que tem por objetivo aumentar o conforto e segurança nas viagens. Estas baseiam-se na partilha de informação entre os vários elementos rodoviários, antecipando perigos. Por exemplo, através dos sensores disponíveis no veículo, pode ser detetada uma avaria que provoque a sua paragem na via. Ao existir um canal de comunicação entre veículos, é possível alertar os veículos mais próximos desse facto, prevenindo uma colisão ou choque em cadeia. Outras aplicações poderão alertar para obras na estrada, acidentes e peões próximos da via. Assim, com base na informação recolhida dos sensores e recorrendo às comunicações veiculares, muitos dos perigos em viagem serão minimizados ou mesmo anulados.

Esta dissertação foca-se na construção, envio e receção de mensagens CAM e DENM (standards europeus) a partir da informação recolhida de vários sensores, garantindo que a

plataforma pode ser usada tanto dentro do veículo – *On-Board Unit* (OBU) – como numa infraestrutura montada ao lado da via – *Road Side Unit* (RSU). Para o primeiro caso (OBU) pretende-se que a plataforma apresente também uma forma de interação com o utilizador.

No desenvolvimento e implementação de sistemas de segurança há que ter em conta vários fatores, entre os quais a fiabilidade da informação e os estreitos requisitos temporais que exigem. A falta de fiabilidade da informação pode levar a erros que conduzem a mais acidentes e mortes na estrada, enquanto que dados desatualizados levam a decisões tardia por parte do sistema.

1.3 Objetivos

O objetivo principal desta dissertação é a implementação de uma camada de suporte que permita que aplicações veiculares possam recolher informação sobre o ambiente em redor, troquem informação com veículos e que façam interface com o utilizador.

Assim, tendo também em conta as normas europeias sobre comunicações veiculares, a plataforma existente e os objetivos do projeto HEADWAY, os objetivos desta dissertação são:

- Estudo da plataforma HEADWAY atual;
- Levantamento das normas sobre comunicações veiculares;
- Escolha do *hardware* e arquitetura que irá suportar as camadas protocolares superiores das normas veiculares;
- Definição do método de recolha da informação e que sensores utilizar, ligação aos sensores do veículo (interface OBD-II);
- Interface com a camada de rede da plataforma;
- Interface com a camada de segurança (*security*) da plataforma;
- Interface com o utilizador;
- Recolha e tratamento da informação com vista construção e envio de mensagens de segurança cooperativa CAM e DENM;
- Construção e envio, receção e processamento de mensagens de segurança cooperativa segundo as normas europeias – mensagens *Cooperative Awareness Message* (CAM) e *Decentralized Environmental Notification Messages* (DENM);
- Interface com uma outra rede auxiliar para gestão remota;

- Integração dos vários módulos em termos de comunicação/fluxo da informação dentro do sistema.
- Implementar testes de onde se possam obter resultados sobre o desempenho do sistema em termos recolha da informação, troca de mensagens CAM e DENM.

1.4 Organização do Documento

O presente documento está organizado da seguinte forma:

- Capítulo 1 – Introdução – esta dissertação é contextualizada, sendo apresentada a introdução, motivação, enquadramento e objetivos;
- Capítulo 2 – Comunicações Veiculares – apresenta-se o estado da arte em termos de aplicações e comunicações veiculares. São abordados os conceitos fundamentais, normas e estado atual sobre o desenvolvimento deste conjunto de tecnologias;
- Capítulo 3 – Camada de Aplicação ITS-G5 – descrição da camada de suporte à aplicação, as suas funcionalidades e as mensagens de rede veicular segundo o ETSI;
- Capítulo 4 – Especificação e Arquitetura do Sistema – abordam-se as funcionalidades que se pretendem implementar, bem como o papel de cada subsistema.
- Capítulo 5 – Implementação da Camada de Aplicação – é apresentada e justificada a implementação efetuada de forma a capacitar o sistema com as funcionalidades definidas no capítulo anterior. Neste capítulo também é explicada a integração que foi feita com outros módulos elaborados por terceiros.
- Capítulo 6 – Avaliação do Sistema – contém a definição dos testes de validação que foram aplicados ao sistema implementado de forma a obter medidas sobre o desempenho do mesmo. Os resultados obtidos são também apresentados e discutidos neste capítulo.
- Capítulo 7 – Conclusão e Trabalho Futuro – encontram-se as conclusões retiradas dos resultados do capítulo anterior. Também neste capítulo são apresentadas linhas para futuros trabalhos que possam ter esta dissertação como base.

Capítulo 2

Comunicações Veiculares

De forma a enquadrar o leitor num dos assuntos principais desta dissertação, neste capítulo serão abordados os conceitos e normas fundamentais em torno de um novo tipo de comunicações sem fios que, através da troca de informação entre veículos, pretendem diminuir os riscos associados ao tráfego automóvel, as Comunicações Veiculares.

Além dos conceitos básicos, como a arquitetura da rede, equipamento básico (*On-Board Unit* (OBU) e *Road Side Unit* (RSU)), alocação de espectro eletromagnético e normas definidas, serão apresentadas em detalhe algumas aplicações que as comunicações veiculares podem proporcionar. Estas, na sua maioria, relacionam-se com a segurança dos veículos e das pessoas que neles se deslocam: segurança passiva, alertando cada condutor para perigos na via (obras, acidentes, mau tempo, viaturas paradas na estrada, etc) e segurança ativa, em que as aplicações detetam esses perigos e atuam diretamente no veículo. São abordadas também outras aplicações não relacionadas com segurança que oferecem, no veículo, informações de trânsito, restaurantes e hotéis na periferia, bem como outros serviços gerais de Internet, acesso ao *email*, por exemplo.

Por fim, fala-se um pouco dos desenvolvimentos nesta área através da apresentação de alguns projetos e produtos comerciais que atualmente existem em torno das comunicações veiculares.

2.1 Introdução

A nível mundial tem-se verificado um aumento do tráfego rodoviário juntamente com o consequente aumento das despesas com este tipo de transporte, entre elas encontram-se as despesas com acidentes [2].

Segundo Moustafa and Zhang [3], uma forma a mitigar estes problemas seria a partilha de informação entre os vários elementos rodoviários de modo a que estes estejam constantemente conscientes do que os rodeia. É neste contexto que surge um novo tipo de comunicações, as comunicações veiculares. Estas são comunicações sem fios que permitem que os vários elementos da estrada – veículos, peões, etc – comuniquem entre si, partilhando informação sobre o ambiente em que se encontram: Por exemplo, a existência de obras na via ou ocorrência de algum acidente. Desta forma, cada agente rodoviário vai agir mais corretamente, prevenindo a ocorrência de incidentes/acidentes.

Nos últimos anos, esta nova classe de comunicações/redes têm vindo a emergir devido aos avanços nas telecomunicações sem fios e aos progressos ocorridos na indústria automóvel, pois atualmente, os veículos que se encontram no mercado já dispõem de diversas tecnologias que aumentam a segurança, entre elas, vários tipos de sensores que auxiliam o condutor. Futuramente pretende-se que a informação proveniente destes sensores seja partilhada entre veículos, através da implementação de uma rede de comunicações veiculares. Estas comunicações poderão, então, ser realizadas entre veículos – *Vehicle to Vehicle* (V2V) – ou entre o veículo e a estrada – *Vehicle to Infrastructure* (V2I) – ou mesmo entre o veículo e outra rede não-veicular (celular, WiFi,..., acesso à Internet), denominadas de comunicações *Vehicle to Anything* (V2X)[4]. Para que isto seja possível tanto o veículo como a estrada terão de ser equipados com novos dispositivos que serão responsáveis por lidar com estas comunicações: no veículo existe um dispositivo denominado OBU e na estrada (infraestrutura rodoviária) existirão dispositivos do tipo RSU.

As aplicações veiculares de segurança assumem um papel principal dentro destas redes, pois são elas que permitem a redução da sinistralidade, mas também poderão existir outras aplicações para outros fins – entretenimento e informação – através uma possível ligação à Internet. Deste modo, durante as viagens, os ocupantes do veículo poderão, por exemplo, consultar o *email*, obter informações sobre pontos de interesse em redor ou mesmo realizar *streaming* de vídeos de alta definição, viajando a elevada velocidade.

Devido a todas estas aplicações e à sua importância existem atualmente vários desenvolvimentos a nível mundial em torno das redes veiculares, sendo um dos principais a definição, em 1999 por parte do IEEE, de normas sobre comunicações rádio de curta distância (DSRC) e sobre comunicações veiculares (WAVE) [5][6]. Estas normas, inicialmente criadas para os Estados Unidos da América (EUA), foram também adotadas, ainda que com as devidas adaptações, na Europa, primeiro pelo C2C-CC e depois, regulamentadas completamente, pelo

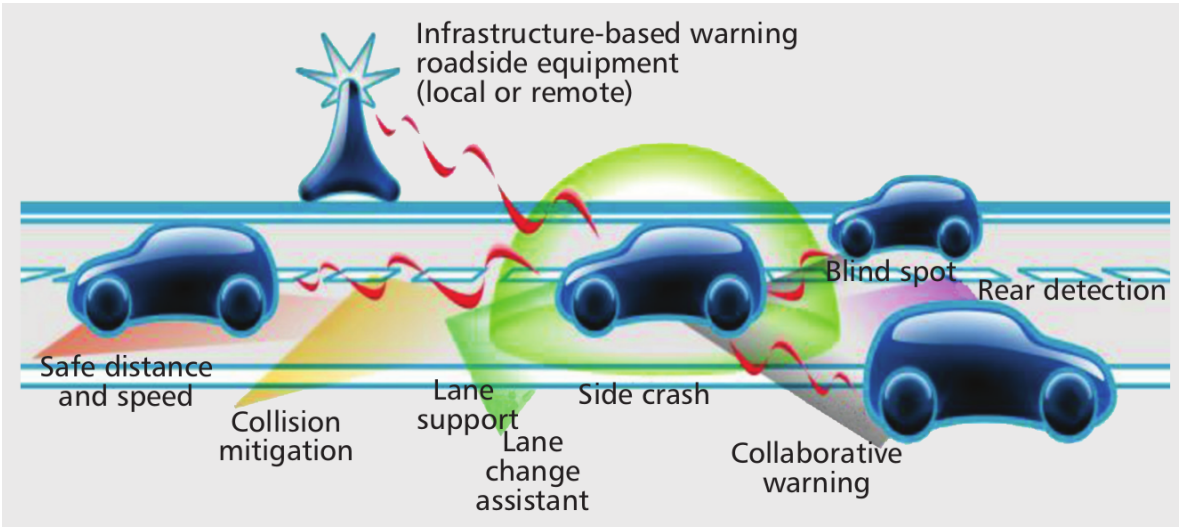


Figura 2.1: Visão geral Comunicações Veiculares (*fonte:* [4])

ETSI [7] [8].

2.1.1 Características

Torna-se evidente, pela descrição em 2.1, que este tipo de comunicações têm requisitos que as diferenciam das restantes. Segundo [9] algumas propriedades das comunicações veiculares são:

- **Muito Baixa Latência:** Crucial para que cada veículo tenha informação atualizada, mesmo quando se desloca a grande velocidade.
- **Fiabilidade:** As falhas na transmissão têm de ser mitigadas ao máximo para não induzir mais erros.
- **Privacidade e Segurança:** Falhas na segurança podem implicar vidas humanas. Privacidade também tem que ser assegurada de forma a que estas redes sejam aceites tanto pela população como pelas autoridades governamentais.

2.1.2 Alocação do Espectro de Frequências

O espectro eletromagnético é um recurso limitado e já bastante ocupado. Foi neste contexto de extrema lotação do espectro que, para as comunicações DSRC, foi atribuída uma pequena banda de frequências em torno dos 5.9MHz. Nos EUA, em 1999, foram alocados 75MHz (5.850 GHz a 5.925 GHz) e na Europa, em 2008, a Comissão Europeia alocou 30MHz

(5875-5905 MHz) [10]. A alocação de espectro para DSRC nas várias regiões do mundo é apresentada na figura 2.2 [5] [11].

Estas bandas, tanto na Europa como nos EUA, são livres mas licenciadas, quer isto dizer que a sua utilização é gratuita, mas obedece a algumas restrições em termos de utilização e potência transmitida.

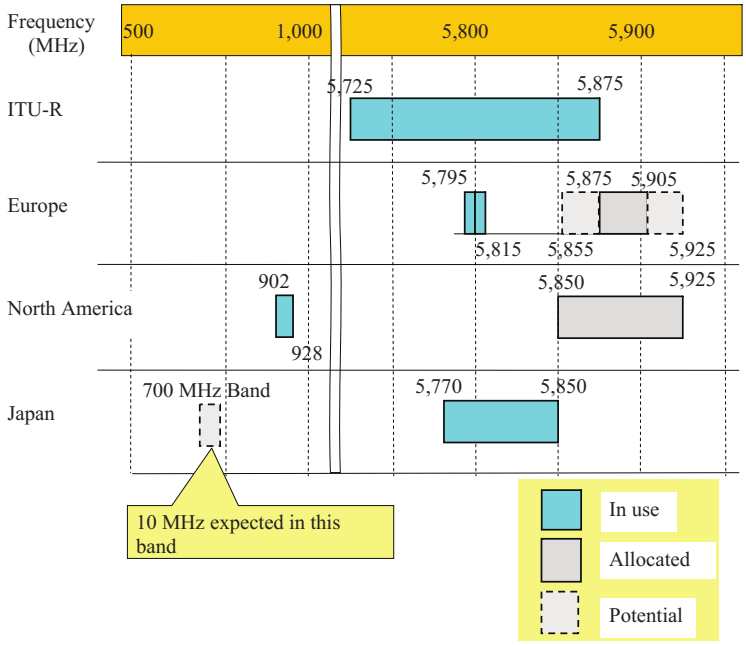


Figura 2.2: Espectro alocado para DSRC nas várias regiões do mundo (*fonte:* [9]).

2.1.3 Arquitetura de Rede

A arquitetura de rede é um aspeto fulcral nas comunicações veiculares. Pois neste tipo de redes, denominadas de *Vehicular Ad-hoc NETwork* (VANET), os vários nós encontram-se em movimento, assim a topologia da rede é altamente dinâmica logo, são grandes os desafios para uma boa transmissão de dados.

Em termos de arquiteturas VANET existem três possibilidades principais [12]:

- Puramente celular/ *Wireless Local Area Network* (WLAN);
- Puramente ad-hoc;
- Arquitetura híbrida.

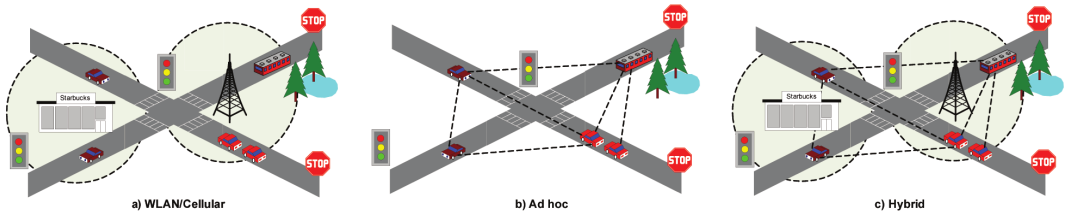


Figura 2.3: Arquitetura VANET (*fonte:* [12])

A primeira trata-se de uma rede baseada numa infraestrutura com pontos de acesso ligados entre si e que oferecem cobertura sem fios aos restantes elementos da rede. Aqui os veículos teriam de usar estes pontos de acesso para comunicarem e se ligarem à Internet. Esta arquitetura apresenta vantagens como a conetividade sempre presente e a ligação a outras redes. No entanto, devido ao elevado custo das infraestruturas, existe o risco de não haver cobertura em certas zonas, impedindo a comunicação entre veículos.

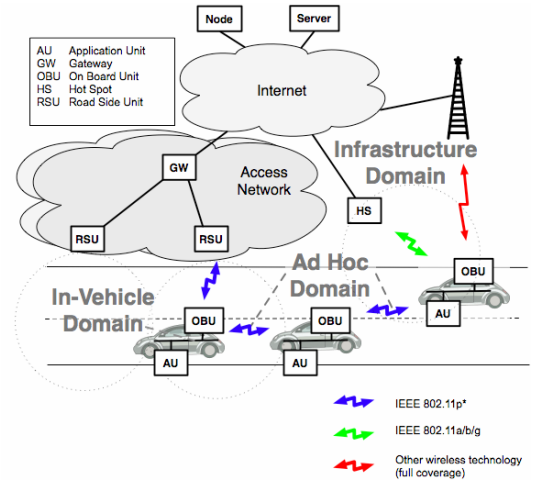
Numa rede puramente ad-hoc os nós da rede (estações ITS) funcionam como *routers* que encaminham a informação entre eles. Esta arquitetura apresenta um custo mais baixo em relação à anterior, uma vez que dispensa o uso de mais infraestruturas além das OBUs dos veículos, mas, como é óbvio, acarreta desvantagens quando existe uma baixa concentração de veículos. A falta de acesso a outras redes é também uma desvantagem.

Por fim existe a arquitetura híbrida. Esta tenta colmatar as desvantagens das duas anteriores já que utiliza comunicação ad-hoc e usa uma infraestruturas fixas – RSUs – colocadas em pontos estratégicos da via. Assim, consegue-se ter uma boa cobertura e ligação a outras redes.

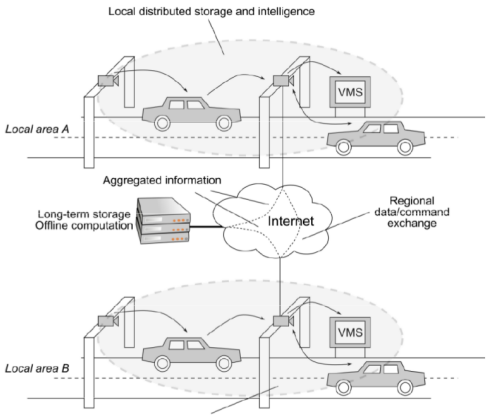
2.1.3.1 Arquitetura C2C-CC

O *Car-to-Car Communication Consortium* representa um conjunto de empresas europeias da indústria automóvel que, em 2007, propôs uma estrutura para redes veiculares com base na arquitetura híbrida descrita anteriormente [13].

Nesta arquitetura cada veículo possuí uma OBU. Esta permite ao veículo comunicar com os que o rodeiam de uma forma ad-hoc mas também usando uma rede infra-estruturada através de RSUs colocadas na beira da estrada. Além disso, esta arquitetura prevê que o OBU do veículo se possa ligar diretamente a *hot-spots* WiFi e a redes celulares *Global System for Mobile Communications* (GSM) e *Universal Mobile Telecommunication System* (UMTS).



(a) Arquitetura C2C-CC (*fonte:* [13])



(b) Arquitetura ICSI (*fonte:* [14])

Figura 2.4: Arquiteturas de Rede em VANETs

2.1.3.2 Arquitetura ICSI

A arquitetura proposta pelo consórcio *Intelligent Cooperative Sensing for Improved traffic efficiency* (ICSI) é em grande parte semelhante à anterior em termos da utilização de OBUs e RSUs e na disponibilização de acesso à Internet, mas bastante diferente em termos de robustez e escalabilidade da rede.

Segundo o *website* do ICSI [14], a arquitetura anterior (Arquitetura C2C-CC) é puramente hierárquica, em que a informação flui das RSUs e OBUs até uma unidade central que gere o tráfego. Esta aproximação não é escalável a um grande número de elementos, não é flexível de forma a suportar o aumento e mudanças nos ITSs e, ainda de acordo a mesma fonte, apresenta problemas de latência e segurança.

Assim, o consórcio ICSI propõe um arquitetura de rede em que a inteligência e poder de atuação se encontra distribuída por vários elementos da rede chamados de *gateways*. Estes dispõem de capacidade de armazenamento de dados e poder de processamento que permitem a execução de aplicações veiculares localmente, colmatando os problemas que a arquitetura anterior apresenta.

2.2 Aplicações para Comunicações Veiculares

Segundo Moustafa and Zhang [3] e como se pode facilmente deduzir, sem aplicações veiculares apelativas e relevantes, as comunicações veiculares não terão sucesso comercial, nem

mesmo pertinência em termos de segurança e eficiência do tráfego. Sendo assim, as aplicações veiculares são um assunto central e que tem merecido bastante atenção também por parte de organismos reguladores como o ETSI, que já definiu um conjunto de aplicações a serem implementadas e distribuídas até 2015, este assunto é abordado mais à frente (ponto 2.4.2).

É aceite que as aplicações para comunicações veiculares podem ser classificadas como [9]:

1. **Aplicações de Segurança Rodoviária**: servem para diminuir a probabilidade de ocorrência de acidentes e perda de vidas, através de avisos e mensagens que têm por objetivo informar o condutor acerca do que se passa em torno do seu veículo.

2. **Aplicações de Eficiência e Gestão do Tráfego**: relacionam-se com a melhoria da fluidez e coordenação do tráfego rodoviário, disponibilizando informações atualizadas sobre o local, mapas e, em geral, mensagens relevantes em tempo real.

3. **Aplicações de Informação/Entretenimento (*Infotainment*)**: Informações mais gerais e de entretenimento:

    - Serviços de cooperação local (pontos de interesse, etc...);
    - Serviços que se podem obter normalmente a partir do acesso à Internet.

Entre estes três conjuntos, as aplicação de segurança e de eficiência do tráfego são as que têm tido mais atenção por parte dos agentes envolvidos, mas aplicações de *Infotainment* ou de conforto também podem acelerar a penetração no mercado uma vez que proporcionam serviços que atraem a generalidade do público, como o acesso à Internet, informação sobre pontos de interesse ou *video-on-demand*.

Os pontos seguintes consistem numa discussão alargada sobre cada um destes conjuntos de aplicações.

2.2.1 Aplicações Ativas de Segurança

As aplicações de segurança tem como principal objetivo evitar acidentes e, consequentemente, diminuir a taxa de mortalidade nas estradas. Para que este objetivo seja cumprido, estas terão de conseguir transmitir a informação de forma fiável, eficiente e determinística. As aplicações de segurança podem ser sub-divididas em *Cooperative Collision Avoidance* (CCA) e em *Emergency Warning Message* (EWM) [5] [3].

2.2.1.1 *Cooperative Collision Avoidance*

As aplicações *Cooperative Collision Avoidance* (CCA) permitem que os veículos comuniquem entre si de forma a evitar colisões. Entre eles circula informação sobre a dinâmica de cada veículo (posição, velocidade, aceleração, rota, etc.) e também informações estáticas (características físicas do veículos: dimensão, tipo, etc.). Perante estas informações cada veículo pode evitar, por exemplo, colisões em cadeia e acidentes por mudanças nas condições da estrada.

2.2.1.2 *Emergency Warning Message*

As aplicações *Emergency Warning Message* (EWM) são usadas para o veículo, quando deteta um acidente ou um perigo na estrada, avisar os veículos vizinhos do tipo de perigo e localização do mesmo. Desta forma, podem-se antecipar as medidas para que o impacto desse incidente seja minimizado. Estas mensagens são disseminadas para/de veículos dentro da zona de relevância do evento, enquanto este persistir.

2.2.2 Aplicações de Eficiência e Gestão de Tráfego

Estas servem para melhorar a coordenação do tráfego, tanto a nível de fluidez do trânsito, como a nível ambiental: redução da poluição rodoviária.

Mais detalhadamente, as aplicações de eficiência e gestão do tráfego têm como objetivo informar o condutor das condições do trânsito na área envolvente ou numa área especifica, podendo até sugerir o melhor percurso. Estas são menos exigentes em termos de latência que as aplicações ativas de segurança (ponto 2.2.1). Tal como as anteriores, as aplicações de eficiência e gestão de tráfego podem-se dividir em aplicações de monitorização de tráfego e aplicações de assistência nas intersecções [5][3].

2.2.2.1 Monitorização de Tráfego

As aplicações desta categoria podem fornecer informações de grande detalhe, localizadas e atualizadas dentro de um raio centrado na localização atual do veículo [3]. Para isso, é assumido que as estações ITS (veículos e infraestruturas de beira de estrada) terão de estar equipadas – ver 2.5 – com um sistema de navegação ou, pelo, menos com um mapa digital com marcos standard. Também é assumido que, em cada veículo, seja possível aceder a sensores relevantes, como velocidade ou aceleração, de forma a aumentar o detalhe da informação. Estes dados são obtidos e formatados ao nível da camada de aplicação e depois

são transmitidos através da rede. Este mecanismo relaciona-se com o que vai ser descrito no ponto 3.2.

2.2.2.2 Assistência nas Intersecções

O objetivo deste tipo de aplicações é, principalmente, a assistência aos condutores nas intersecções, pois, de acordo com Moustafa and Zhang [3], as colisões entre veículos poderão ser evitadas se estes comunicarem entre si. Deste modo, num futuro próximo, pode-se pensar em *embutir os semáforos em cada veículo* – semáforos virtuais [15]. Por fim, este sistema anti-colisões pode ser estendido a peões através da inclusão de aplicações ITS em *smartphones*. Estas poderão utilizar as comunicações de curta distância, como o Bluetooth, para avisar os veículos da aproximação de um peão [5].

2.2.3 Aplicações de Conforto

Como o nome indica, aplicações de conforto servem para tornar as viagens mais agradáveis para os passageiros. Os serviços oferecidos por este tipo de aplicações englobam a reprodução de filmes, música ou acesso à Internet, também poderá ser disponibilizada informação sobre pontos de interesse próximos: restaurantes, hotéis, monumentos, etc.

Atualmente, existem construtores automóveis que oferecem acesso à Internet através da rede 3G/4G [3], mas o objetivo será a utilizar a rede veicular para este efeito, uma vez que oferecem, em teoria, maiores garantidas de qualidade de serviço devido à maior abundância de pontos de acesso (em pontos com pouco tráfego automóvel, RSUs terão de ser colocadas perto da via para colmatar falhas de cobertura).

Para desenvolver estas aplicações existe a obrigação de obedecer a certas normas impostas pelos organismos reguladores. Este assunto será abordado nas próximas secções: secção 2.3 (*Normas IEEE WAVE - EUA*) e secção 2.4 (*Normas ETSI ITS e ISO CALM - Europa*).

2.3 Normas IEEE WAVE - EUA

Este ponto abordará as normas (*standards*) em torno das DSRCs nos EUA, ou seja, o conjunto de normas WAVE constituído pelos standards IEEE 802.11p e IEEE 1609.1-4 que foram definidos pelo IEEE.

Apesar do trabalho desenvolvido nesta dissertação não se basear nos standards WAVE, é importante introduzir-lo porque serve de base às normas europeias publicadas pelo ETSI

(ponto 2.4).

2.3.1 Camadas Protocolares

Segundo Karagiannis et al. [9], os standards que compõem a pilha protocolar WAVE dizem respeito às diferentes camadas protocolares do modelo *Open Systems Interconnection* (OSI) (figura 2.5), sendo eles os seguintes:

IEEE 802.11p

Trata da camada física *PHYsical Layer* (PHY) e *Medium Access Control* (MAC) de forma a que a norma geral IEEE 802.11 possa operar num ambiente veicular. Para isso foram definidas, no 802.11p, duas novas entidades, a *Physical Layer Management Entity* (PLME), que gere a camada física, e a *MAC Layer Management Entity* (MLME) que gere a camada MAC.

IEEE 802.2

Especifica a camada lógica de controlo - LLC (*Logical Link Control*).

IEEE 1609.4

Responsável operação em múltiplos canais de frequência.

IEEE 1609.3

Fornece os serviços de encaminhamento e endereçamento da informação que são exigidos pelas normas WAVE na camada de rede e de transporte. Aqui foram definidos dois protocolos: o WSMP e o comum IPv6. Segundo Hartenstein et al. [5], as aplicações de segurança deverão utilizar, principal e prioritariamente, o WSMP, mas aplicações não relacionadas com esse aspeto também o poderão usar, todavia, é suposto que utilizem o protocolo *Internet Protocol version 6* (IPv6) ao nível da camada de rede e *Transmission Control Protocol* (TCP) ou *User Datagram Protocol* (UDP) na camada de transporte.

IEEE 1609.2

Define os serviços relacionados com a segurança e privacidade na troca de informação.

IEEE 1609.1

Fornece uma aplicação que permite que uma OBU, com recursos limitados e processos complexos de funcionamento, os possa processar, em segurança, numa unidade de processamento remota.

SAE J2735

Define parte das camadas superiores do modelo OSI WAVE, nomeadamente estrutura das mensagens a ser trocadas pelos vários elementos das estrada.

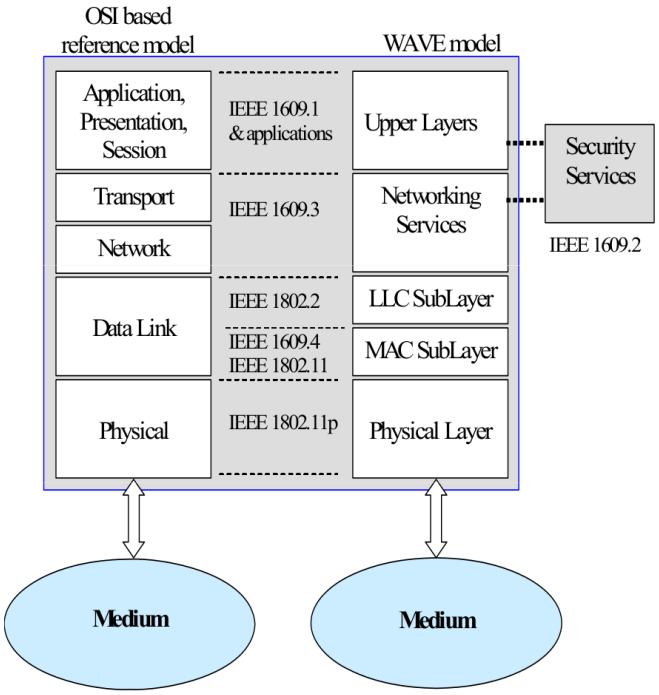


Figura 2.5: Pilha protocolar das normas WAVE (*fonte:* [9])

Depois desta visão geral sobre as normas WAVE, em seguida, abordar-se-á com mais pormenor a normas IEEE 802.11p, IEEE 1609.3 e SAE J2735 pois são as mais relevantes neste trabalho: a primeira porque é a tecnologia de rádio que suporta a comunicação, a segunda porque se relaciona diretamente com a camada de aplicação e terceira porque é a base das aplicações ITS cooperativas.

2.3.2 WAVE PHY e MAC: IEEE 802.11p

Segundo Uzcategui and Acosta-Marum [16], a norma IEEE 802.11p foi criada a partir da alteração da IEEE 802.11a de modo a que esta funcione em ambientes veiculares, tendo em conta os seguintes aspetos:

- Operação a maior distância (cerca 1km);
- Condições de elevada mobilidade e velocidade;

- Ambiente em que predomina o efeito de *Multipath* - existência de vários *ecos* do sinal do sinal recebido;
- Grande número de redes *ad hoc* sobrepostas e onde se exige uma elevada qualidade de serviço (QoS);
- A natureza das aplicações veiculares.

Assim, o standard IEEE 802.11p apresenta [17]:

1. **Aumento dos requisitos na máscara de transmissão:** Máscaras de espectro mais restritivas.
2. **Melhoramento dos requisitos impostos ao recetor:** Aumento da capacidade de rejeição de canais adjacentes, portanto, reduzindo *cross-talking* entre canais.
3. **Canais com 10MHz de largura de banda:** Com esta largura de canal garante-se que o intervalo de guarda é suficiente para que não haja interferência entre símbolos.
4. **Aumento da frequência de operação para a banda dos 5.9GHz:** Zona do espectro não-paga, mas licenciada, em que se tem de transmitir dentro de certos limites de potência. Este aumento permite que se, mais facilmente, se modifique os dispositivos rádio a trabalhar nos 5GHz para operarem nos 5.9GHz.

2.3.3 Rede e Transporte: IEEE 1609.3

Segundo Uzcategui and Acosta-Marum [16], o standard IEEE 1609.3 define as camadas de rede e transporte do modelo da figura 2.5.

Nesta camada, como foi referido anteriormente (2.5), a arquitetura WAVE suporta dois protocolos: o tradicional IPv6 e o WSMP, que operam em cima de uma única camada *Logical Link Control* (LLC). Esta configuração dupla acomoda comunicações de alta-prioridade que são exigentes em termos de tempo de resposta (através do WSMP), bem como comunicações menos prioritárias e com menos requisitos (através de UDP/TCP/IPv6).

O protocolo WSMP permite que a aplicação envie mensagens curtas e que controle certos parâmetros da transmissão rádio de forma a maximizar a probabilidade das partes envolvidas receberem as mensagens em tempo útil. No entanto, o WSMP não tem capacidade suficiente para suportar as aplicações típicas de Internet, daí a inclusão do IPv6.

O IEEE 1609.3, em termos de funcionalidade, pode ser dividido em dois conjuntos [16]:

- **Serviços de Informação**: Têm a função de transportar a informação (WSMP e UDP/TCP/IPv6);
- **Serviços de Gestão**: Desempenham as funções de configuração e manutenção do sistema: registo das aplicações, gestão do *WAVE Basic Service Set* (WBSS), monitorização da utilização dos canais, configuração do IPv6, monitorização do indicador de potência recebida do canal e manutenção da base de gestão da informação.

2.3.4 Aplicação: SAE J2735

A norma SAE J2735, publicada pelo *Society of Automotive Engineers* (SAE), define parte da camada de aplicação da *stack* WAVE, nomeadamente, a estrutura das mensagens trocadas entre elementos da rede.

Estas mensagens têm o propósito de fornecer às aplicações veiculares um meio de troca de informação entre veículos, permitindo: aplicações que evitam a colisão entre veículos ou aplicações destinadas a informar sobre o trânsito ou mesmo aplicações que fornecem acesso básico à Internet, entre outras. Para isso, a estrutura de cada mensagem foi definida no formato *Abstract Syntax Notation One* (ASN.1) que permite interoperabilidade entre sistemas, uma conversão eficiente entre estruturas de dados e permite a extensão/adição de novos campos de dados. Antes da mensagem ser passada à camada de rede, o *standard* SAE J2735 requer que se aplique a codificação *Distinguished Encoding Rules* (DER) a todas as mensagens enviadas, isto permite que a informação seja representada numa forma mais compacta, reduzindo o número de bits de cada mensagem e, portanto, a transmissão torna-se mais eficiente [5].

Nesta norma, segundo Hartenstein et al. [5], um dos tipos de mensagens mais importante é o tipo BSM. Cada mensagem deste tipo contém informações sobre o estado do veículo que a envia: a sua posição, tamanho e dinâmica. Além disso, a estrutura BSM definida oferece a flexibilidade necessária para que sejam enviadas informações adicionais. As BSMs funcionam como um sistema de *broadcast* em que cada veículo tem consciência da sua vizinhança, isto torna-se fundamental, por exemplo, em termos de prevenção de colisões: o veículo conhece a posição e trajetória dos que o rodeiam e assim, comparando com a sua posição e trajetória, poderá evitar que ocorra uma colisão – ou através de um aviso ao condutor ou mesmo atuando diretamente no veículo, corrigindo a velocidade, trajetória, etc.

Como foi dito as mensagens BSMs pretendem ser uma forma de os veículos se monitorizarem. Segundo a norma SAE J2735, esta monitorização deverá ser feita com cada cada veículo

transmitir mensagens BSM a uma frequência de 10Hz, a não ser que exista um congestionamento do canal, nesse caso, a frequência de transmissão pode ser menor.

As normas apresentadas nesta secção foram as primeiras a regulamentar as comunicações veiculares. Posteriormente, o ETSI apresentou o seu conjunto de normas para a Europa, com algumas diferenças, mas ainda assim baseadas nas do IEEE. Na secção seguinte são apresentadas essas normas.

2.4 Normas ETSI ITS e ISO CALM - Europa

Para a Europa o *International Organization for Standardization* (ISO) e o ETSI ITS têm publicado um conjunto de *standards* relacionados com transportes inteligentes, alguns destes standards/normas dizem respeito a comunicações veiculares.

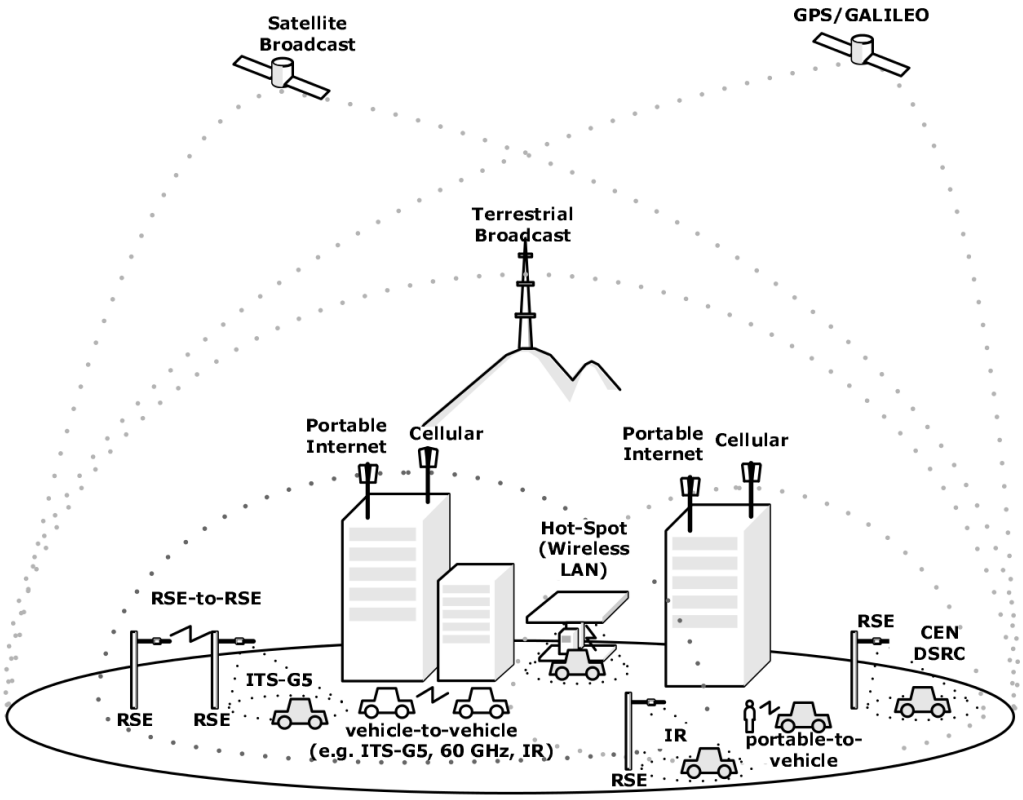


Figura 2.6: Arquitectura Geral CALM (*fonte:* [18])

Atualmente, o uso as redes móveis é elevado (WLAN; *Worldwide Interoperability for Microwave Access* (WIMAX); redes celulares – GSM e UMTS; *Global Positioning System* (GPS), ...) e em qualquer lugar da Europa é possível aceder a uma ou mais redes deste tipo. É

dentro desta filosofia "sempre *on-line*"que se insere o conjunto de *standards Communications Access for Land Mobiles* (CALM). Este combina os sistemas veiculares com a capacidade de ligação a outras redes (figura 2.6). Desta forma, com base nas redes disponíveis, os veículos têm a possibilidade de escolher a melhor combinação de tecnologias de acesso aos vários serviços, tanto de segurança como de *infotainment*.

Segundo Brickley et al. [19], as tecnologias de redes móveis a serem integradas são:

- **WLAN e WIMAX** - Conectividade local e regional;
- **Redes Celulares UMTS e GSM** - Conectividade Global;
- **Satélite e *Broacasting*** - ex. GPS;
- **Redes *Ad Hoc* de Médio Alcance** - DSRC, Infravermelhos e *Millimeter Wave Networks* (MM-Wave).

Com base no cenário representado na figura 2.6, o ETSI definiu a pilha protocolar que serve de referência para as comunicações veiculares na Europa [18].

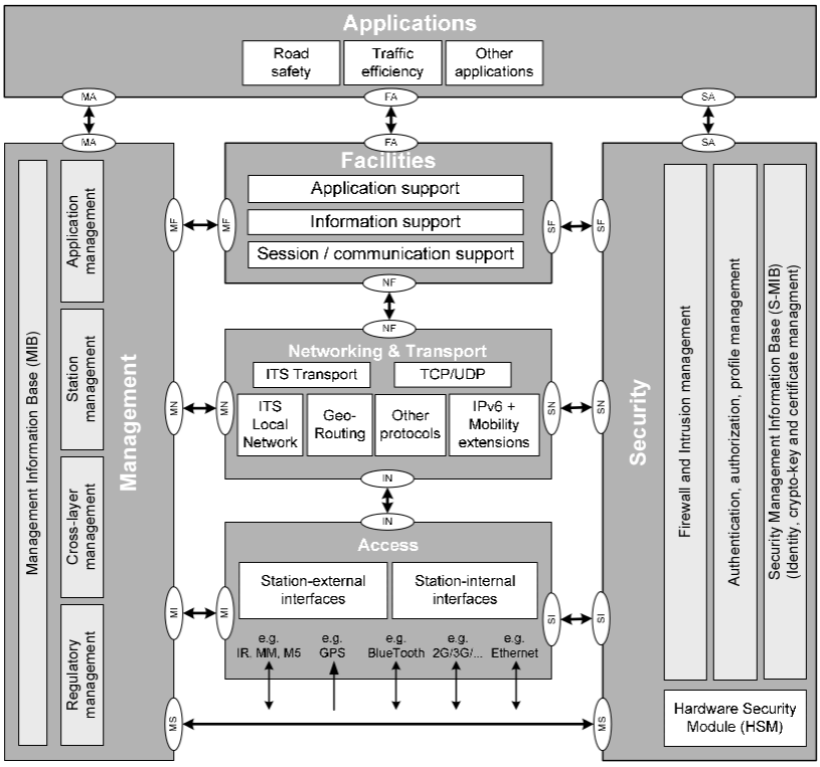


Figura 2.7: Pilha protocolar ETSI ITS (*fonte:* [18])

Na figura 2.7 é apresentada a *stack* protocolar para ITS na Europa. Aqui uma das

principais diferenças em relação às normas Norte Americanas é o facto de cada estação ter à sua disposição vários tipos de redes de acesso (parte mais baixa da pilha protocolar).

O ETSI [18] prevê também que as redes veiculares possam ser utilizadas por peões, ciclistas, etc, para isso foram definidas os seguintes tipos de estações/elementos de rede: *central ITS* (Sistema central de controlo das comunicações), *vehicle ITS* (Sistema colocado em automóveis, camiões, etc., quer estejam em movimento ou estacionados), *roadside ITS* (Sistemas de beira de estrada que auxiliam a condução) e a *personal ITS*. A estação *personal ITS* pode ser implementada, por exemplo, na forma de uma aplicação em dispositivos *hand-held*, ou seja, PDA's e Smartphone's. Logo, caso um peão esteja a atravessar a passadeira, os veículos mais próximos poderão ser avisados desse facto, entre outras aplicações possíveis.

2.4.1 ETSI ITS *stack*

Como se pode ver na figura 2.7, o modelo OSI da arquitetura ITS Europeia é composto por várias camadas, cada uma das quais com funções e requisitos específicos.

Em seguida, é apresentada uma visão geral das funções de cada camada protocolar segundo o que está especificado na norma EN ETSI 302 665 [18]:

Acesso (*Access Layer*)

Resumidamente, esta camada tem funções que permitem o acesso ao meio, tendo que suportar um ou mais canais de comunicação lógicos. Esta camada engloba as subcamadas PHY e *Data Link Layer* (DLL). Esta última ainda engloba a camada MAC e LLC.

Além disso, a *Access Layer* também contém interfaces com as camadas vizinhas – rede e transporte (*Networking and Transport*), gestão (*Management*) e Segurança (*Security*).

Rede e Transporte (*Networking and Transport*)

Esta camada é responsável pela gestão da rede e gestão do transporte da informação. Tendo em conta a arquitetura geral exemplificada na figura 2.6, esta camada suporta os protocolos de rede enunciados a seguir mas, não está restrita a estes, pois, segundo ETSI [18], outros poderão ser adicionados:

- GeoNetworking;
- IPv6 puro e IPv6 sobre GeoNetworking;
- CALM FAST.

Também existe suporte para vários protocolos de transporte:

- UDP;
- TCP;
- Protocolos específicos ITS a definir.

Facilities

A camada *Facilities* trata de fornecer, de uma forma transparente, recursos e funções de suporte à camada de aplicação e ao resto do sistema, nomeadamente:

- Suporte de Aplicação (*Application support*);
- Suporte de Informação (*Information Support*);
- Suporte de Sessão (*Session Support*).

Ou seja, esta camada é responsável por recolher informação que vem da rede e de outras fontes (como sensores) e disponibilizá-la a cada aplicação. Além disso, cada aplicação recorre a esta camada para se autenticar no sistema e ter a possibilidade de efetuar comunicações através das redes disponíveis nas camadas inferiores.
Tendo em conta que os objetivos desta dissertação prendem-se, em grande parte, com esta camada protocolar, no capítulo 3 (*Camada de Aplicação ITS-G5*), esta será abordada com mais detalhe.

Application

Nesta camada funcionam as aplicações ITS. Estas foram agrupadas em Segurança Rodoviária (*Road Safety*), Eficiência de Tráfego (*Traffic Efficiency*) e outras. Como são classes com diferentes importâncias, as duas primeiras terão de ter, em relação às outras, uma maior exigência em termos de fiabilidade, segurança e latência.

Management

É uma entidade que tem a função de gerir todas as restantes camadas da pilha protocolar (figura 2.7).

Security

Esta camada fornece os serviços de segurança e privacidade a todas as camadas da pilha protocolar ETSI ITS.

Por fim, falta mencionar que os interfaces entre as várias camadas protocolares da *stack* estão a ser definidos no conjunto de *standards* ETSI TS 102 723 e por isso não estão disponíveis. Fica a nota de que a previsão para publicação dos mesmos é Dezembro de 2013 [20].

2.4.2 ETSI *Basic Set of Applications*

Em relação à camada de aplicação, o ETSI, de modo a impulsionar a tecnologia neste sector, definiu em TR ETSI. 102 638 v1. 1.1 [21] uma janela temporal entre 2012 e 2015 para implementação deste conjunto de aplicações veiculares básicas – *Basic Set of Applications* (BSA) (tabela 2.1). Estas têm por base a rede veicular 802.11p mas, mais uma vez, prevêm o uso das redes celulares (2G, 3G e 4G) e de *broadcast* (DAB, T-DMB, DVB).

Como se pode verificar pela tabela 2.1, existem três grandes grupos de aplicações que se podem relacionar com os grupos descritos nos pontos 2.2.1 (aplicações de segurança), 2.2.2 (monitorização/ eficiência de tráfego) e 2.2.3 (conforto). Na terminologia ETSI, estes grupos são designados, respetivamente, por:

- ***Cooperative Road Safety*** Visam melhorar a segurança na estrada através da cooperação entre veículos.
- ***Cooperative Traffic Efficiency*** Aplicações para aumentar a fluidez do tráfego.
- ***Cooperative Local Services and Global Internet Services*** Serviços comerciais ou não-comerciais que podem ser *Infotainment* ou conforto. Os serviços baseados na cooperação entre veículos são oferecidos pela própria infraestrutura ITS, enquanto que o serviço de acesso à Internet terá que incluir um fornecedor de Internet externo.

Finalmente há a referir que este conjunto de aplicações básicas foram selecionadas tendo em conta várias entidades relacionadas com os transportes: desde as empresas até aos utilizadores/condutores, passando pelas instituições governamentais europeias [21].

2.5 Hardware: Equipamento de Suporte

Nesta secção irão ser abordados os elementos de *hardware* básicos que estão presentes neste tipo de comunicações, mais especificamente, os componentes necessários ao desenvolvimento e implementação de OBUs.

De acordo com Moustafa and Zhang [3], é assumido que uma OBU terá que apresentar os seguintes elementos:

- **CPU** - implementa as aplicações e os protocolos de comunicação;
- ***Wireless Transceiver*** - envia e recebe informação das estações vizinhas;
- **Recetor GPS** - fornece informações sobre posição e sincronização temporal;

Tabela 2.1: ETSI *Basic Set of Applications* (*fonte:* [21])

Applications Class	**Application**	**Use Case**
Active road safety	Driving assistance - Co-operative awareness	Emergency vehicle warning
		Slow Vehicle Indication
		Intersection collision warning
		Motorcycle approaching indication
	Driving assistance - Road Hazard Warning	Emergency electronic brake lights
		Wrong way driving warning
		Stationary vehicle - accident
		Stationary vehicle - vehicle problem
		Traffic condition warning
		Signal violation warning
		Roadwork warning
		Collision risk warning
		Decentralized floating car data - Hazardous location
		Decentralized floating car data - Precipitations
		Decentralized floating car data - Road adhesion
		Decentralized floating car data - Visibility
		Decentralized floating car data - Wind
Cooperative traffic efficiency	Speed management	Regulatory/ contextual speed limits notification
		Traffic light optimal speed advisory
	Co-operative navigation	Traffic information and recommended itinerary
		Enhanced route guidance and navigation
		Limited access warning and detour notification
		In-vehicle signage
Co-operative local services	Location based services	Point of Interest notification
		Automatic access control and parking management
		ITS local electronic commerce
		Media downloading
Global internet services	Communities services	Insurance and financial services
		Fleet management
		Loading zone management
	ITS station life cycle management	Vehicle software/ data provisioning and update
		Vehicle and RSU data calibration.

- **HMI** - Sistema de *input/output* de informação entre o utilizador e o sistema OBU.

Segundo outro autor (Hartenstein et al. [5]), além dos dados de localização que se podem obter do recetor de GPS, outros dados podem ser obtidos do veículo diretamente através do interface *On-Board Diagnostics* (OBD). Daí podem ser extraídos valores relacionados com a velocidade, temperatura e estado geral de funcionamento do veículo.

Em termos mais específicos, o dispositivo deverá apresentar as seguintes funcionalidades:

- **Interface *Controller Area Network* (CAN)**: para ligação aos vários sensores dos veículos;
- **Interface ou módulo de GSM/UMTS**: para efeitos de monitorização, etc;
- **Módulo GPS/GALILEO**: efeitos de localização (exceto quando o veículo já contém esta funcionalidade, aí utiliza-se o interface CAN para este propósito);
- **Módulo Bluetooth**: para conexão a um *smartphone* (por exemplo), para aplicações de interface com o utilizador. (o mesmo método de cima poderá ser aplicado se o veículo já apresentar esta funcionalidade);
- **Módulo WLAN 802.11**: para ligações de médio alcance e/ou interface com o utilizador através de um *smartphone*.

2.5.1 Requisitos Técnicos

A utilização de dispositivos eletrónicos em veículos, exige que estes apresentem alguma robustez na construção e nas ligações a periféricos uma vez que precisam de resistir a vibrações, turbulências, trepidações e, idealmente, também resistir a acidentes rodoviários que possam ocorrer [22]. Nesta última situação estas OBU poderiam desempenhar, nos veículos, a mesma função que as *caixas negras* desempenham nos aviões, isto é, fornecer informações sobre as condições em que um acidente ocorreu para que futuramente se possa prevenir esses acontecimentos. Em Kargl et al. [22] é também referido que os veículos são produtos com um tempo de vida relativamente elevado, durando bastantes anos. Assim as OBUs têm de estar preparadas para durar, sem falhas de maior, bastante tempo e, estratégias de *update*, terão de ser implementadas partindo sempre do princípio que o condutor não tem quaisquer conhecimentos de eletrónica ou informática, isto é, terão de ser totalmente automáticas e não poderão pôr em risco a segurança dos veículos e dos ocupantes.

Para além de características físicas, estes módulos terão de cumprir requisitos em termos de desempenho em várias aplicações. Atrasos e erros podem levar ao mau funcionamento do

veículo, erros de condução (colisões..) e, consequentemente, danos humanos e materiais [22].

Segundo Karagiannis et al. [9], os módulos que sirvam de suporte a aplicações veiculares têm que garantir tempos de *latência* relativamente baixos:

- menor que 100 ms para as aplicações ativas de segurança rodoviária e para aplicações de eficiência e gestão do tráfego;
- entre 100 ms e menos de 200 ms para aplicações de cooperação local.
- até de 500 ms para aplicações relacionadas com o acesso geral à Internet.

Entre as aplicações veiculares, uma das mais exigentes em termos de desempenho do sistema será *streaming* de vídeo entre veículos, em que a OBU poderá desempenhar o papel de cliente ou servidor nesse serviço.

Por exemplo, o projeto *Distributed Routing and Infotainment through VEhicular Inter-Networking* (DRIVE-IN), implementa um serviço de *streaming* de video entre veículos que permite que um veículo tenha acesso às imagens recolhidas por câmaras instaladas em outro veículos, de forma a que o condutor tenha uma visão global do que se passa á sua volta [23][24] [25]. Logo, é também importante que o dispositivo tenha alguma capacidade de processamento e reprodução de vídeo e áudio.

Para além disso, há que ter em conta que o dispositivo tem que correr várias aplicações simultaneamente, isto é, o facto de se estar a fazer um *streaming* do vídeo da câmara do carro da frente não pode impedir que o dispositivo alerte o condutor, em tempo útil, da ocorrência de um acidente. Assim, terá de possuir alguma memória *Random Access Memory* (RAM) e, também, será mais prudente usar um CPU com vários núcleos de processamento devido às várias aplicações a correr ao mesmo tempo.

No entanto, um dos requisitos que sempre se impõe é o preço final do módulo (quando mais baixo ele for, melhores serão as probabilidades de vingar no mercado automóvel) e assim, este também é um fator a ter em conta na escolha do *hardware* para uma OBU/RSU.

2.6 Projetos ITS

Existe um grande interesse em torno das comunicações veiculares. Podem-se encontrar projetos relacionados com este tema um pouco por todo o mundo.

2.6.1 EUA

Segundo uma pesquisa efetuada em Karagiannis et al. [9], os EUA foram dos primeiros países a preocupar-se em aumentar a inteligência nos transportes de forma a reduzir a sinistralidade e aumentar a fluidez do tráfego. Este trabalho teve início em 1991, com o governo a criar o programa *Intelligent Vehicle Highway Systems* (IVHS). Decorridos alguns anos, foi decidido que se iria usar redes sem fios baseadas no IEEE 802.11, levando à criação do conjunto de standards WAVE que incluía o 802.11p. Este standard, atualmente, serve de base ou está de alguma forma incluído nas comunicações veiculares em todo o mundo.

Nos dias de hoje, os projetos de maior importância em curso no EUA são:

- **IntelliDrive(sm):** dos testes efetuados no âmbito deste projeto derivaram bastantes recomendações e linhas de orientação que visaram vários aspetos das comunicações veiculares. Em primeiro lugar testou-se o grau de adequação das normas WAVE ao ambiente veicular, tendo-se apontado os pontos fortes e os pontos a melhorar nestas comunicações, como a necessidade de se incluir um método de medição da qualidade de sinal. Posicionamento das estações ITS, segurança, mensagens de aviso e fiabilidade da informação são outros assuntos que este projeto trata e sobre os quais tece recomendações que ajudam a implementar comunicações mais eficientes e adaptadas ao ambiente veicular.
- **Vehicle Safety Communications (VSC):** este é um consórcio que se preocupa com os requisitos de desempenho das aplicações de segurança ITS. Entre os requisitos mais importantes que foram recomendados estão: as mensagens de segurança devem ter uma latência máxima de 100 ms, devem ser geradas e enviadas 10 mensagens deste tipo por segundo e deverão poder ser transmitidas a uma distância mínima de 150 metros.

Comparando estes requisitos e recomendações com os standards que estão atualmente em vigor – secção 2.3.1 e 2.4 – pode-se concluir acerca da grande importância e influência que estes projetos tiveram no panorama atual das comunicações veiculares.

2.6.2 Japão

No Japão, o assunto das comunicações veiculares surgiu mais tarde, em 1996, com o programa *Comprehensive Plan for ITS in Japan*. Neste país, são usadas normas próprias. Por exemplo, é usada uma gama de frequências na banda dos 5.8 GHz, ao contrário da Europa e dos EUA que usam a banda dos 5.9GHz (2.2).

Atualmente existem projetos que se têm especializado em várias áreas das comunicações veiculares no Japão [9]:

- **Smartway:** Usa a banda dos 5.8 GHz e suporta cobrança de portagens, pagamentos eletrónicos, bem como informações e avisos acerca do tráfego. Este sistema foi demonstrado com sucesso em 2006 e tornou-se operacional no verão desse ano.
- **ASV (Advanced Safety Vehicle):** Este projeto foca-se na tecnologia de suporte a comunicações entre os veículos e entre veículos e a infraestrutura. O seu objetivo final é a criação de aplicações rodoviárias para aumentar a eficiência e segurança do tráfego.
- **ITS-Safety 2010:** Responsável por definir as frequências a serem usadas para comunicações veiculares, tanto V2V como V2I.

2.6.3 Europa

Na Europa, a investigação e desenvolvimento na área das comunicações veiculares é impulsionada, principalmente, pela comissão europeia. Esta relaciona-se com vários outros órgãos e consórcios europeus – figura 2.8 – com vista à implementação de um sistema que permita aos cidadãos europeus usufruírem de uma maior fluidez no trânsito, maior segurança e menor poluição.

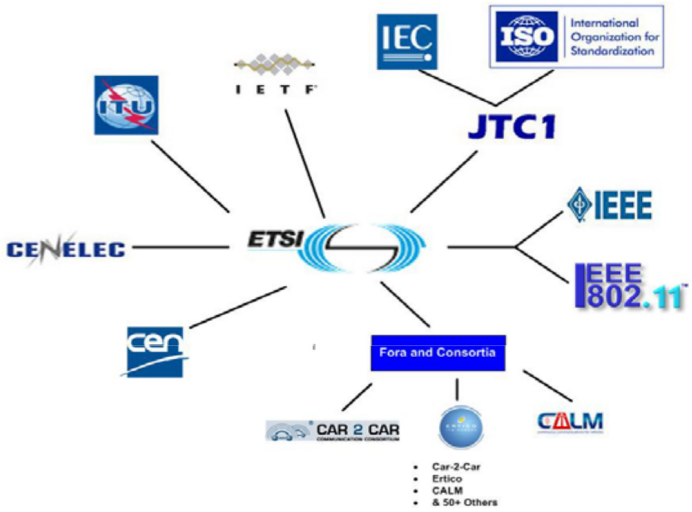


Figura 2.8: Relação entre órgãos de regulação europeus (*fonte:* [9])

Além do organismo principal e responsável pelas normas europeias – o ETSI – existe, no campo das comunicações veicular, outra entidade que se destaca, o *Car-to-Car Communication Consortium*. Este, como referido superficialmente em 2.1.3, é um consórcio que engloba

grandes empresas da industria automóvel, tecnologia, bem como universidades e centro de investigação europeus de referência [26]. Este consórcio é responsável pela arquitetura e princípio geral das comunicações veiculares na Europa [13].

Outros projetos importantes, segundo Karagiannis et al. [9], são:

- **Cooperative Vehicle Infrastructure Systems (CVIS)** No âmbito do CVIS foram desenvolvidas aplicações como sugestão de rotas mais rápidas e de alerta de veículo de emergência. Durante 2011 foram conduzidos testes em vários países europeus.
- **Network on Wheels (NoW)** Forneceu soluções para mecanismos de *routing* e protocolos de encaminhamento com base na localização geográfica, adaptação de comunicações sem fios para ambientes veiculares reais, soluções para antenas veiculares, segurança em comunicações ad-hoc e melhoramentos na segurança e rapidez das comunicações entre veículos.

Outro projeto europeu relativamente recente, e onde esta dissertação se insere é o *Intelligent Cooperative Sensing for Improved traffic efficiency*. Este projeto tem como objetivo diferenciador a construção de um sistema ITS com a inteligência distribuída de forma a que possa ser escalável, flexível e, além disso, seguro e resistente a falhas [27].

Uma das entidades que participa neste consórcio é o Instituto de Telecomunicações de Aveiro, o qual teve como projeto pioneiro o HEADWAY que consiste no projeto e implementação de hardware e software que permita um tráfego rodoviário mais eficiente [28].

Capítulo 3

Camada de Aplicação ITS-G5

No modelo OSI comum a camada protocolar superior é a camada de aplicação, que é constituída por todas as aplicações de um sistema. Mas, na pilha protocolar ETSI a camada de aplicação foi partida em duas: a parte superior que se continua a chamar camada de aplicação e a parte inferior designada de camada de suporte à aplicação (ITS *Facilities*). Como o nome indica, esta camada fornece recursos às várias aplicações ITS, como recolha e suporte de informação e suporte para autenticação. Além disso, é responsável pela gestão das mensagens que permitem a colaboração entre veículos – CAM e DENM. Através da monitorização destas mensagens, as aplicações ITS na camada superior recebem a informação da existência de perigos na via e poderão exibir alertas ao condutor ou mesmo atuar no veículo.

Resumindo, a camada de suporte é de extrema importância na pilha protocolar ETSI uma vez que gere um conjunto de recursos que possibilitam, de uma forma simples, a implementação de aplicações ITS com funcionalidades avançadas. Neste capítulo irão ser apresentadas as funcionalidades e características gerais desta camada e, em mais pormenor, as mensagens da rede veicular que esta camada deverá ser capaz de gerir.

3.1 Características Gerais

No contexto desta dissertação a camada *ITS Facilities* é a mais importante, pois é nela que se gere a informação, se dá suporte à camada de aplicação e se gerem as mensagens de alto nível que serão transmitidas na rede veicular.

Esta camada é apresentada em detalhe na figura 3.1 e, como se pode verificar, ela é constituída por vários blocos, cada um com funções específicas [18]:

Suporte Básico HMI

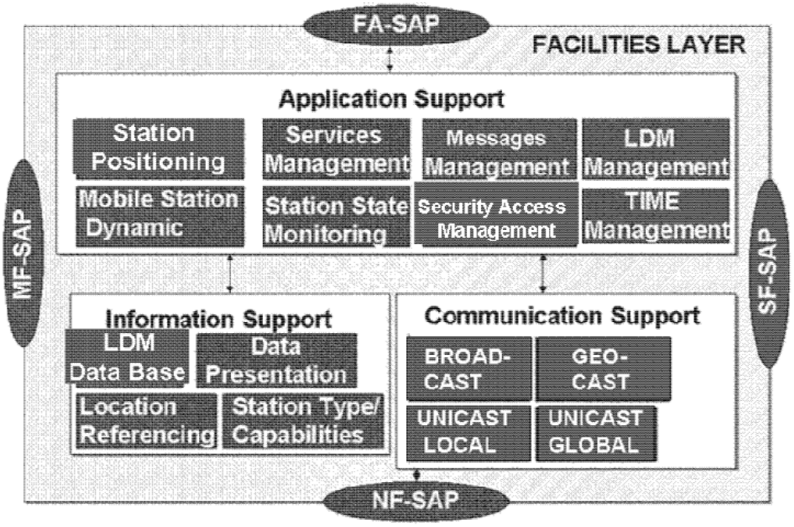


Figura 3.1: ITS *Facilities Layer* (*fonte:* [18])

Suportar interface com o utilizador/condutor.

Posição e Tempo

Fornecer informações de posição geográfica (latitude, longitude, altitude) e uma referência temporal.

Manutenção do sistema

Disponibilizar um meio de instalação de novo software, bem como atualizar o existente.

Gestão dos recursos da estação ITS

Gerir informações como o tipo de estação (veículo, RSU,...), informações estáticas ou dinâmicas relacionadas com a estação.

Cruzamento de Informação

Fazer o cruzamento de dados provenientes de várias fontes e disponibilizar os melhores ao sistema.

Gestão de mensagens

Gerar, enviar e encaminhar mensagens CAM, DENM e mensagens de anúncio de serviços.

Assim, esta camada engloba funções e características essenciais no desenvolvimento de qualquer aplicação veicular, entre as quais a gestão de mensagens, que é abordada na secção seguinte.

3.2 Mensagens CAM e DENM

Entre as estações ITS existe troca de mensagens. Isto permite que, sobre este mecanismo, funcionem várias aplicações de ajuda ao utilizador, como por exemplo, prevenção de acidentes/incidentes rodoviários. Assim, a norma EN ETSI 302 665 [18] prevê a existência de vários tipos de mensagens, entre as quais:

- Mensagens Periódicas: CAM - *Broadcast* do estado da estação ITS que envia a mensagem;
- Mensagens de Eventos: DENM - Anúncio às estações vizinhas da ocorrência de um determinado evento.

Estas mensagens antes de serem disseminadas na rede têm de ser codificadas segundo *Unaligned Packed Encoding Rules* (UPER) [29][30][31]. Á semelhança das DER, estas regras permitem que estruturas de dados definidas no formato ASN.1, como é o caso das CAMs e DENMs, sejam compactadas, ou seja, sejam representadas num menor número de bits, o que torna a transmissão mais eficiente [31].

Em seguida, serão abordadas as mensagens CAM (3.2.1) e DENM (3.2.2): Estrutura, informação que contêm, *timings*/requisitos para o seu envio. A informação completa sobre todos os campos CAM pode ser encontrada, em formato ASN.1, nos anexos, já que existem estruturas de dados comuns entre estes dois tipos de mensagem.

3.2.1 CAM

Segundo ETSI [30], as *Cooperative Awareness Message*s (CAMs) são um tipo de mensagens enviadas periodicamente para os nós da rede vizinhos (*Single-Hop*). Mais especificamente, são distribuídas na rede ITS-G5 e fornecem informações sobre a presença, posição e estado de cada elemento. Assim, todas as estações ITS que participam na rede têm de ser capazes de gerar, enviar e receber estas mensagens. Desta forma, cada estação ITS terá consciência do ambiente que a rodeia: posição, movimento e dados dos sensores dos vizinhos.

Estas mensagens são geridas na camada *ITS Facilities* – ponto 3 – e trata-se de uma funcionalidade obrigatória numa estação ITS.

3.2.1.1 Funcionamento Geral

O gestor de mensagens, na camada *ITS Facilities* – figura 3.1 – gera a mensagem a partir da informação disponibilizada pelos módulos *time management*, *station state monitoring* e *mobile*

station dynamic e em seguida passa a mensagem para a camada inferior de forma a que esta seja transmitida na rede. Por outro lado, ao receber uma CAM, o gestor de mensagens terá de a validar e, basicamente, passá-la para uma base de dados georeferenciada, denominada por *Local Dynamic Map* (LDM). Esta base de dados será a responsável por manter um conjunto de informações que representam a vizinhança da estação ITS – automóveis, peões, semáforos, acidentes.

3.2.1.2 Requisitos Temporais e Regras para Construção e Envio de Mensagens CAM

Ainda segundo o ETSI [30], dependendo do *use case* que se está a tratar, existem diferentes requisitos em termos de *timming* no envio de mensagens CAM – Tabela 3.1.

Tabela 3.1: CAM *Use Cases* (*fonte:* [30])

Use Case	Frequência Min (Hz)	Latência Min (ms)
Emergency Vehicle Warning	10	100
Slow Vehicle Indication	2	100
Intersection Collision Warning	10	100
Motorcycle Approaching Indication	2	100
Collision Risk Warning	10	100
Speed Limits Notification	1 a 10	100
Traffic Light Optimal Speed Advisory	2	100

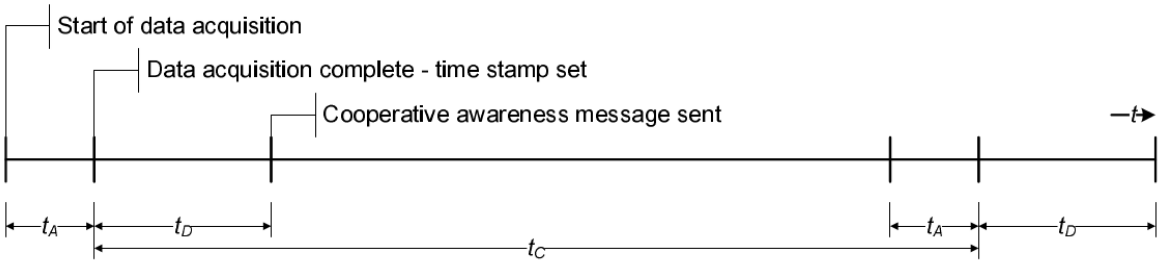


Figura 3.2: Requisitos Temporais CAM (*fonte:* [30])

Em suma, as mensagens CAMs terão de ser geradas e passadas para as camadas inferiores com um intervalo máximo de 1s (1Hz) e um intervalo mínimo de 0.1s (10Hz). O sistema deve também garantir que tanto a construção (t_A) como o envio da cada mensagem para a rede (t_D) não excede os 50ms. Assim, segundo a figura 3.2, $t_A \leq 50ms$ e $t_D \leq 50ms$.

Finalmente, além destes requisitos e dos *use cases* descritos na tabela 3.1 existe a obrigatoriedade de enviar mensagens CAM segundo as seguintes regras:

- Intervalo máximo de geração e envio de CAMs: 1s;
- Intervalo mínimo de geração e envio de CAMs: 0.1s;
- Gerar CAM quando a diferença entre o valor absoluto atual e o anterior da direção da estação ITS for maior que 4º.
- Gerar CAM quando a diferença absoluta entre a posição atual e a anterior for maior que 5 metros;
- Gerar CAM quando a diferença absoluta entre a velocidade atual e a anterior for maior que 1 metro/s;
- Estas regras são verificadas a cada 100 ms.

3.2.1.3 Estrutura das Mensagens

A estrutura das CAMs foi definida em TS ETSI 102 637-2 v1.2.1 e pretende englobar o maior número possível de estações ITS [30]. Nesta norma também é apontado um conjunto de perfis de estações ITS principais, para cada um destes, descreve-se os campos obrigatórios e opcionais que a respetiva estação ITS deve preencher ao construir uma CAM. Apesar das diferenças, todas as mensagens geradas devem incluir um cabeçalho comum com a identificação da versão do protocolo usado (*protocolVersion*), uma marca temporal (*generationTime*) e tipo de mensagem (*messageID*). Outros parâmetros relacionados com a identificação da estação, localização e movimento também são comuns a todas as CAM de todos os perfis.

Os perfis presentes em TS ETSI 102 637-2 v1.2.1 são:

basicVehicle

Este perfil é usado principalmente por veículos privados e serve de base à maior parte dos restantes perfis.

- Caraterísticas: móvel, privado e com relevância física;
- Campos Obrigatórios: `vehicleType`, `vehicleSpeed`, `vehicleSpeedConfidence`, `heading`, `headingConfidence`, `stationLength`, `stationWidth`, `curvature`, `curvatureConfidenc` `longAcceleration`, `posConfidenceEllipse`, `exteriorLights`, `accelerationControl`, `confidenceStationLength/confidenceStationWidth`, `crashStatus`;

- Campos Opcionais: `distanceToStopLine`, `turnAdvice`, `occupancy`, `dooropen`, `curvatureChange`.

basicIRS

O perfil basicIRS é uma RSU. Esta é instalada na infraestrutura rodoviária e, consequentemente, não é fisicamente relevante pois não apresenta perigo para o tráfego rodoviário.

- Caraterísticas: estacionário, não é privado e sem relevância física;
- Campos Obrigatórios: nenhum além dos presentes em todas as mensagens;
- Campos Opcionais: (nenhum).

emergencyVehicle

Usado em veículos de emergência.

- Caraterísticas: móvel, não é privado e com relevância física;
- Campos Obrigatórios: os mesmos que "basicVehicle"mais emergencyResponseType;
- Campos Opcionais: `lightBarInUse`, `sireneInUse`.

publicTransportVehicle

Usado em veículos de transporte público.

- Caraterísticas: móvel, não é privado e com relevância física;
- Campos Obrigatórios: os mesmos que "basicVehicle"mais `PublicVehicleType` e `DoorOpen` (desde quando uma porta se abre até 30s depois de ela se fechar);
- Campos Opcionais: os mesmos que "basicVehicle"mais `PTLineDescription`, `trafficLightPriori` `scheduleDeviation`, `occupancy`.

A falta de informação para preencher as estruturas de dados obrigatórias não deve impedir que a mensagem seja enviada. Isto é, a mensagem deve ser enviada para a rede mesmo quando não existe, naquele momento, informação para a construir completamente.

3.2.2 DENM

Ao contrário das CAM que têm de ser enviadas periodicamente, as mensagens DENM só são enviadas quando a estação ITS deteta eventos que possam pôr em causa a segurança das pessoas. Isto é, as DENMs são usadas em aplicações relacionadas com *Road Hazard Warning* (RHW). Estas aplicações são baseadas em eventos e compostas por múltiplos *use*

cases – ver tabela 3.2 – que vão definir a estrutura final da mensagem a enviar e a forma como as estações vizinhas devem proceder para evitar ou minimizar os efeitos desse evento [31].

3.2.2.1 Funcionamento Geral

Estas mensagens são geradas pelo gestor de mensagens a partir dos módulos *time management*, *station state monitoring* e *mobile station dynamic* (figura 3.1) e ainda a partir da base de dados georeferênciada *Local Dynamic Map* (LDM). Quando construída a mensagem é passada para as camadas inferiores.

O procedimento geral para se lidar com este tipo de mensagens é o seguinte [31]:

- Ao detetar um evento que corresponde a um *use case* RHW conhecido, a estação ITS começa a fazer o *broadcast* da DENM respetiva para as estações localizadas dentro da área geográfica a que o evento diz respeito;
- A transmissão dessa mensagem é repetida com uma certa frequência;
- Este *broadcast* persiste enquanto o evento existir;
- O *broadcast* da DENM termina de duas maneiras: automaticamente depois de um determinado tempo ou, então, quando uma estação ITS envia uma DENM especial a reportar que o evento deixou de existir.
- As estações ITS que recebem as DENM processam a informação e decidem o tipo de notificação/aviso que vão mostrar ao utilizador.

3.2.2.2 Estrutura da Mensagem

Uma mensagem DENM, além do ITS *header* que todas as mensagens devem ter, apresenta-se dividida em 3 partes principais (figura 3.3):

DENM management container

Contém informação para gestão da DENM, ou seja, grau de confiança, evolução e termino do evento. Além disso, esta informação permite distinguir, claramente, entre as diferentes estações ITS que originam as mensagens, bem como os diferentes eventos ocorridos.

DENM situation container

Informação que descreve o evento detetado, assim como o seu potencial impacto na

Tabela 3.2: DENM: *Use Cases*, Condições de desencadeamento e término (*fonte:* [31])

Use Case	**Condição Desencadeamento**	**Condição de Termino**
Emergency Electronic Break Light	Hard breaking of a vehicle	Automatic after the expiry time
Wrong Way Driving Warning	Detection of a wrong way driving by the vehicle being in wrong way direction	Vehicle being in the wrong way has left the road section
Stationary vehicle - accident	e-Call triggering	Vehicle involved in the accident is removed from the road
Stationary vehicle - problem	Detection os a vehicle breakdown or stationary vehicle with activated warnings	Vehicle is removed from or has left the road
Traffic Condition Warning	Traffic Jam Detection	End of traffic jam
Signal Violation Warning	Detection of a vehicle being violating a signal	Signal violation corrected by the vehicle
Road Work Warning	Signalled by a fix or moving roadside ITS sattion	End of the roadwork
Collision Risk Warning	Detection of a turning collision risk by a roadside ITS station	Elimination of the collision risk
	Detection of a crossing collision risk by a roadside ITS station	Elimination of the collision risk
	Detection of a merging collision risk by a roadside ITS station	Elimination of the collision risk
Hazardous location	Detection of a hazardous location	Automatic after the expiry time
Precipitation	Detection of a heavy rain or snow by a vehicle (activation of the windscreen wrappers)	Detection of the end of the heavy rain or snow situation
Road adhesion	Detection of a slippery road condition (ESP activation)	Detection of the end of the slippery road condition
Visibility	Detection of a low visibility condition (activation of some lights or antifog)	Detection of the end of the low visibility condition
Wind	Detection of a strong wind condition (stability control of the vehicle)	Detection of the end of the strong wind condition

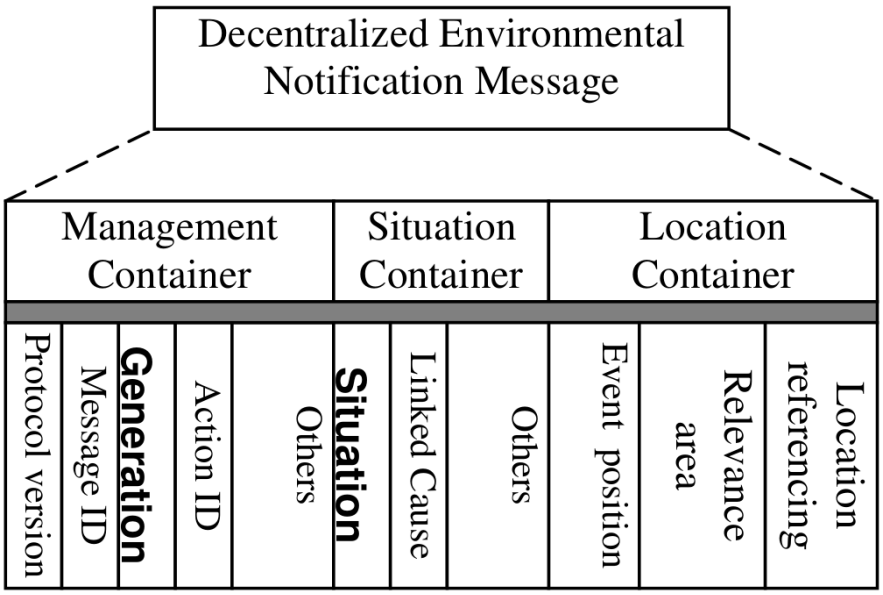


Figura 3.3: Estrutura de uma mensagem DENM (*fonte:* [31])

segurança e fluxo rodoviário. Este tipo de informação está associada aos *use cases* da tabela 3.2. Segundo o ETSI [31], existem as seguintes estruturas de dados principais: *Traffic flow effect*, *Cause code*, *Sub cause code*, *Linked cause code*, *Severity* e *Basic event characteristics*. Se a estação ITS tiver mais informação relevante disponível, esta pode adicioná-la à mensagem.

DENM location container

Contém informação relacionada com os aspetos geográficos do evento detetado. Especificamente existem três estruturas de dados principais:

- *Event Position*: Informa sobre a posição exata onde o evento ocorreu e a extensão (área) que o evento cobre.
- *Relevance Area*: É a área a que o evento ocorrido diz respeito. ou seja, as DENMs devem ser disseminadas para todas as estações que se encontrem dentro desta área geográfica.
- *Location referencing*: Itinerário (ou vários itinerários) que podem levar à localização do evento.

Concluindo, este tipo de mensagens pode conter uma grande quantidade de informação. Como seria praticamente impossível uma única estação ITS recolher todos os dados de uma vez, o *standard* prevê a utilização de um mecanismo de atualização das DENMs. Assim,

uma outra estação que detete o mesmo evento pode atualizar a mensagem anterior com nova informação relevante de forma a que haja o máximo de informação sobre o evento para as estações vizinhas. Por outro lado, este mecanismo de atualização de mensagens permite que a duplicação de mensagens sobre o mesmo evento seja mínima: as mensagens que tenham uma versão antiga vão sendo descartadas até desaparecerem da rede.

No capítulo seguinte são abordadas as funcionalidades, especificação e arquitetura do sistema que permite a recolha de informação, construção, envio e receção de mensagens do tipo DENM e CAM.

Capítulo 4

Especificação e Arquitetura do Sistema

Neste capítulo é abordada a especificação, arquitetura e funcionalidades do sistema englobado por esta dissertação. Este pretende implementar parte das camadas protocolares superiores da *stack* para comunicações veiculares ETSI. Consequentemente, a sua especificação segue a estrutura que este organismo propõe, isto é, uma divisão em duas camadas protocolares distintas: camada de suporte à aplicação e camada de aplicação.

A primeira camada, de suporte à aplicação, tem com função principal fornecer meios que permitam à camada de aplicação aceder a dados de sensores, dispositivos HMI, mensagens que circulam na rede e também segurança de forma simples, abstraindo a complexidade inerente ao acesso a estes recursos. Por outro lado, a camada de aplicação usa esses meios de forma a controlar o fluxo de informação no sistema, ou seja, obtenção de dados dos sensores, construção e envio de mensagens CAM e DENM e interface com o utilizador.

A arquitetura de sistema escolhida pretende ser modular e de fácil compreensão para que durante a sua implementação e teste seja simples fazer o seu *debug* e manutenção. Assim, na camada de suporte á aplicação existem três módulos principais:

- **Sensores**: recolha de dados de sensores (GPS, OBD-II, etc) e interface HMI;
- **Informação**: cruzamento de dados;
- **Mensagens**: Gestão de mensagens da rede (CAM e DENM) recorrendo aos módulos de segurança para encriptar as mensagens e ao módulo de rede para troca de mensagens com outras estações.

Por sua vez, a camada de aplicação é responsável por monitorizar e controlar todos estes processos utilizando o interface de suporte à aplicação.

Por fim, referir que esta dissertação utiliza e comunica com outros módulos (figura 4.1) de *software* desenvolvidos para o projeto HEADWAY:

- Aplicação de deteção de acidentes e interface com utilizador para smartphone – Victor Gomes;
- Módulo de encriptação de dados baseado em IEEE 1609.2 – Daniel Duarte;
- Módulo de rede WSMP – Paulo Sousa.

Estes módulos foram desenvolvidos no âmbito das respetivas dissertações de mestrado, em 2013.

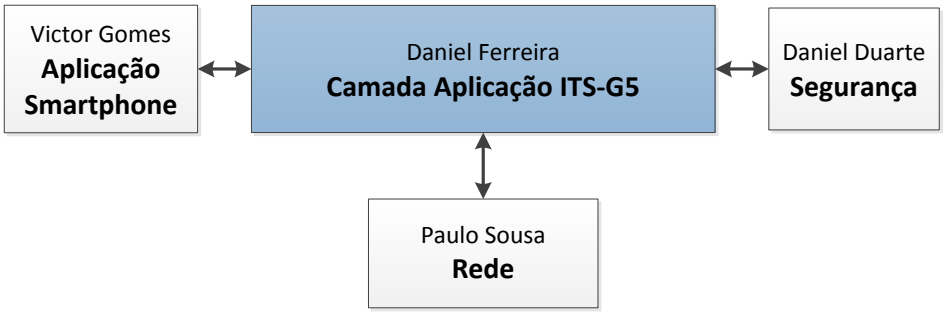


Figura 4.1: Dissertações diretamente relacionadas com esta dissertação.

Este capítulo encontra-se organizado em três secções. Na primeira secção introduz-se o capítulo, na secção seguinte descrevem-se as funcionalidades que o sistema deve apresentar e na última secção (terceira) é finalmente apresentada a arquitetura do sistema.

4.1 Introdução

Como foi dito anteriormente, as aplicações ITS são de extrema importância para a redução de acidentes e para o aumento do conforto nas viagens e, sendo assim, o ETSI definiu 2015 como o prazo para implementação de um conjunto de aplicações de segurança básicas descritas em TR ETSI 102 638 v1.1.1 [21].

Este projeto tem, então, a finalidade de desenvolver um protótipo de acordo com os *standards* para comunicações veiculares do ETSI para a camada de suporte da aplicação –

ITS Facilities (ponto 3) – e de uma aplicação que possa demonstrar a utilidade e validade destas aplicações num contexto aumento da segurança rodoviária.

No ponto 3 (*Camada de Aplicação ITS-G5*) verificou-se que esta camada terá que oferecer um conjunto de funcionalidades complexas mas, como se trata de um primeiro protótipo, escolheu-se começar pelas mais determinantes na ótica de um sistema veicular cooperativo, como o envio de mensagens CAM e DENM. De acordo com a pesquisa efetuada sobre estas mensagens – ponto 3.2 – para as construir há que ter acesso a vários sensores (posicionamento, condições do veículo, etc), bem como à rede 802.11p para que estas sejam enviadas e recebidas. A aplicação em si terá de controlar o envio destas mensagens e ser responsável pela segurança e privacidade destas, utilizando um módulo de encriptação / desencriptação. Além disso, terá de dispor de uma forma de HMI de modo a que o utilizador possa interagir com o sistema. A figura 4.2 representa o esquema simplificado do que se pretende implementar.

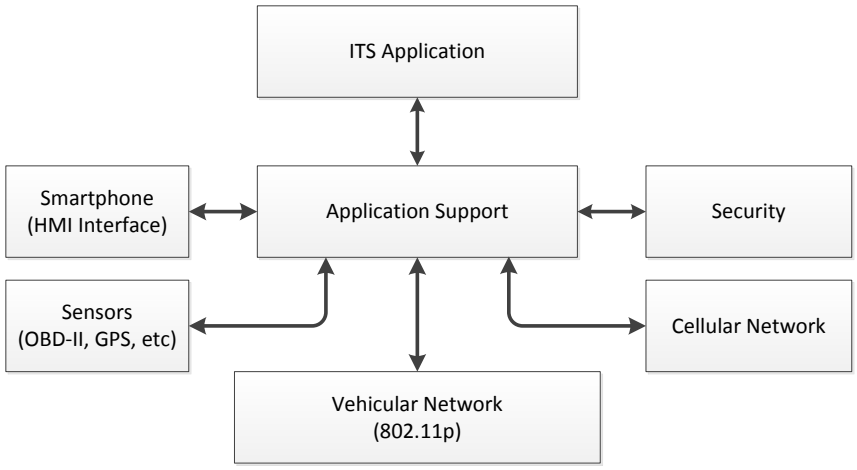


Figura 4.2: Visão geral do sistema OBU / RSU

4.2 Funcionalidades

Com vista a cumprir os objetivos a que esta dissertação se propõe, a camada de suporte à aplicação deverá conter as funcionalidades descritas em seguida, a maior parte delas resulta dos requisitos exigidos pelas normas ETSI ITS que regem este tipo de aplicações (3.2).

Posicionamento (GPS)

Fornecer dados relativos ao posicionamento, recorrendo ao sistema GPS ou a outras fontes de dados de posicionamento;

Estado do Atual Veículo

Dados sobre o estado atual do veículo. Obtidos, principalmente, através do interface de diagnóstico do próprio (interface OBD-II);

Rede de Backup

Comunicar com a rede celular de modo a utilizá-la como rede de backup, monitorização e atualização.

Comunicação com a camada de Rede

Enviar e receber de informação através da rede veicular.

Comunicação com a camada de Segurança

Segurança e privacidade, encriptação da informação.

Mensagens CAM e DENM

Primitivas para construção e validação de mensagens CAM e DENM de acordo com as normas.

Gestão de Mensagens CAM e DENM

Processar as mensagens recebidas e enviadas, bem como a definir do *timing* de envio das mesmas.

Interface para a camada de aplicação

Expor uma interface de comunicação com a camada de aplicação de forma a que esta possa ter acesso e controlar as funções mais importantes do sistema.

Por outro lado, pretende-se que a aplicação ITS, baseando-se na camada de suporte, execute as seguintes funções:

Suporte para HMI

Interface entre o utilizador e o sistema, com alertas e outras informações relevantes à condução.

Monitorização

Monitorização remota do tráfego pode ser importante para efeitos otimização e fluidez do mesmo. Esta funcionalidade encontra-se aqui para efeitos de demonstração e prova de conceito, uma vez que levanta questões de privacidade.

Deverá também existir a possibilidade de ativar ou desativar estas funcionalidades de forma a que o mesmo sistema possa ser usado tanto como OBUs como RSUs.

4.3 Arquitetura do Sistema

Analisando as funcionalidades definidas em 4.2, percebe-se que algumas delas são complexas e englobam outras "sub-funcionalidades de suporte". Além disso, de forma a seguir o standard europeu em relação à camada de suporte à aplicação e à camada de aplicação, estas funcionalidades deverão ser divididas por camadas e em termos de tarefas mais simples para que o resultado final seja também simples e modular. Assim, de modo e endereçar esses requisitos, optou-se pela arquitetura representada na figura 4.3.

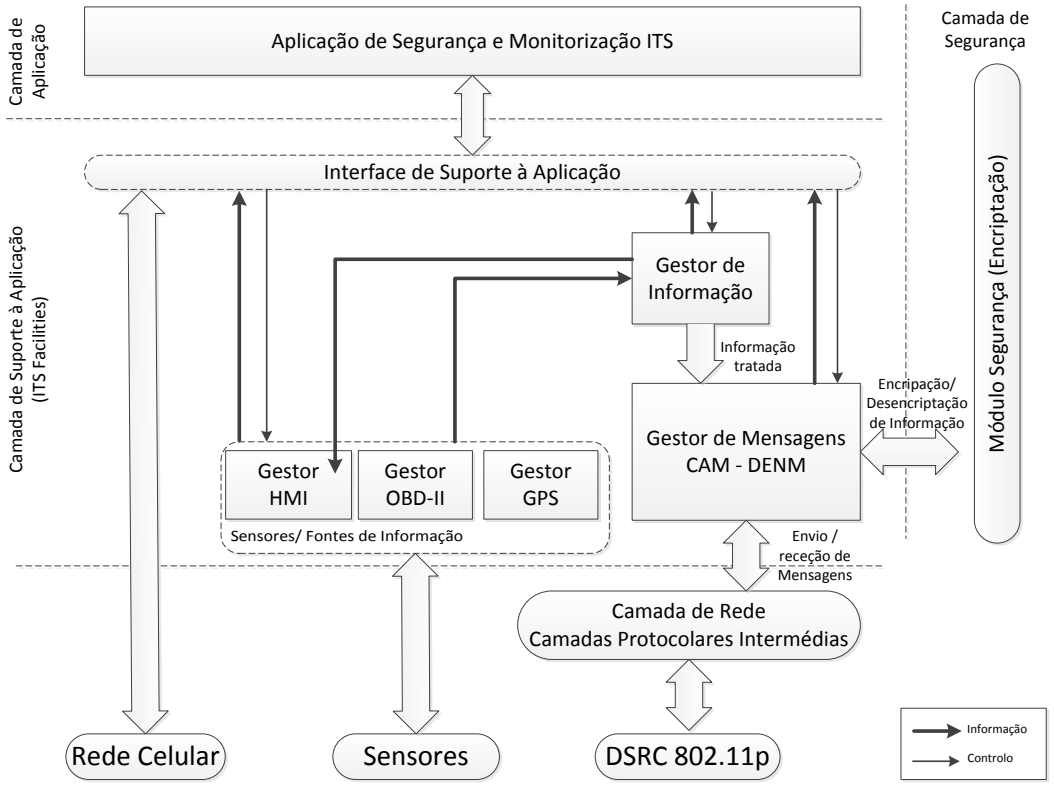


Figura 4.3: Relação entre os blocos fundamentais do sistema.

Nessa figura são visíveis os vários módulos que constituem o sistema e a interação entre eles. Em termos gerais, a aplicação usa o Interface de Suporte à Aplicação (ISA) oferecido pela camada inferior para implementar as funcionalidades descritas em cima. Por outro lado, a camada de suporte terá que conseguir recolher informação e fornecer meios para que esse

interface responda da forma que se espera. A camada de suporte também deverá conter entidades que recolham e disponibilizem a informação a partir dos sensores (Gestor HMI, Gestor GPS, Gestor OBD-II), uma entidade que trate e cruzem a informação das várias fontes disponíveis (Gestor de Informação) e uma entidade que faça a gestão das mensagens trocadas com a camada de rede (Gestor de Mensagens). Para efeitos de monitorização o módulo "Rede Celular"deverá conseguir gerir os pedidos de acesso à rede 3G/4G por parte da aplicação.

Em seguida são explicadas as funções e relações entre os vários blocos, fora desta explicação ficam os módulos de segurança e camada de rede, uma vez que não fazem parte do espectro desta dissertação.

4.3.1 Sensores/Fontes de Informação

Este elemento, responsável pela comunicação com os sensores, engloba três entidades:

Gestor (GPS)

Responsável pela obtenção de dados de posicionamento através do dispositivo recetor GPS. Para isso, terá que reportar se o dispositivo está fisicamente ligado ao sistema ou não. Se sim, configurar a ligação e começar de imediato a recolher e a disponibilizar ao sistema dados de posicionamento, como latitude, longitude, altitude, velocidade, etc.

Gestor OBD-II

Deverá obter dados relevantes dos sensores do veículo, principalmente velocidade e estado de avaria mas se possível, outros dados relevantes como aceleração (vertical e horizontal), portas abertas, ocupação, etc. Para isso, este gestor deverá tratar da ligação ao dispositivo e, logo que exista um ligação válida, disponibilizar os dados dos sensores ao sistema.

Gestor HMI

Este elemento não pode ser considerado um simples gestor de sensores pois a comunicação terá de ser bidirecional. Este módulo recebe *inputs* do condutor sobre o ambiente em redor (acidentes avistados, por exemplo) e envia para o anterior dados recolhidos, avisos e alertas de perigos.

Através dos sensores supracitados deverá ser recolhido o máximo de informação de forma a que, por exemplo, as mensagens CAM e DENM possam ser o mais completas possível.

4.3.2 Gestor de Informação

A finalidade desta entidade é cruzar e tratar os dados provenientes das várias fontes de informação e disponibilizá-los ao sistema, principalmente ao gestor de mensagens CAM-DENM, tendo em conta as questões de concorrência no acesso a esses dados.

4.3.3 Gestor de Mensagens

Este gestor é responsável por, a partir da informação obtida pelo gestor de informação, construir, validar, encriptar (recorrendo ao módulo de segurança) e enviar mensagens CAM e DENM de acordo com o standard – *ver ponto 3.2*. Por outro lado, é responsável pela receção, validação (desencriptar usando módulo de segurança) e disponibilização das mensagens recebidas da camada de rede.

4.3.4 Rede Celular

Deverá fornecer funcionalidades que permitam o estabelecimento de uma ligação de dados à rede celular (GSM, UMTS, *Long Term Evolution* (LTE)) para efeitos de atualização, *backup* e monitorização.

4.3.5 Interface de Suporte à Aplicação

O Interface de Suporte à Aplicação (ISA) é o interface que a camada de suporte expõe à camada de aplicação. Observando o diagrama da arquitetura de software proposta (figura 4.3), este deverá permitir:

- Acesso direto aos dados dos sensores;
- Acesso à informação contida no gestor de informação;
- Acesso ao estado da comunicação com os diversos periféricos;
- Controlo e monitorização dos elementos/entidades da camada de suporte.

Este controlo quase total da aplicação de segurança sobre os elementos da camada inferior permitirá, nesta fase de protótipo, um melhor conhecimento sobre o que ocorre no sistema e oferece vantagens em termos de *debug* de elementos isoladamente.

No capítulo seguinte é abordada de a implementação de cada um destes módulos.

Capítulo 5

Implementação da Camada de Aplicação

5.1 Introdução

Este capítulo pretende apresentar a forma como este sistema foi implementado. O que se pretende é então um sistema que funcione como OBU ou como RSU. Este deverá ter implementadas as funcionalidades referidas em 4.2 e ser modular de forma a isolar cada subsistema para deteção e correção de erros, aplicação em cenários diferentes, bem como acrescentar novas funcionalidades facilmente.

Em termos de hardware o trabalho elaborado focou-se, principalmente, na definição da arquitetura base. Por motivos de simplicidade, flexibilidade e custo escolheu-se usar um *Single Board Computer* (SBC) – o Raspberry Pi – como unidade de processamento de todos os dados que circulam de/para os periféricos.

Estes periféricos, que comunicam com o SBC através do interface *Universal Serial Bus* (USB), estão compreendidos em três categorias: *Human Machine Interface* (HMI), Sensores e Rede. Nesta implementação introduziu-se um *smartphone* Android que pretende tratar da interação com o utilizador – HMI principal. Quanto a sensores, existe o leitor OBD-II (leitura de sensores do veículos), o recetor GPS (posicionamento) e o *smartphone* que, além de HMI, disponibiliza ao sistema os dados de cada um dos sensores que contém. O *smartphone* também funciona como dispositivo de deteção de acidentes, através de uma aplicação para o efeito, desenvolvida fora do âmbito desta dissertação. Na categoria de rede inserem-se o *dongle* USB de ligação a uma rede de *backup* (3G/4G) e o dispositivo desenvolvido no âmbito do projeto

HEADWAY que permite a ligação à rede veicular ITS.

O *software* desenvolvido também seguiu a lógica da modularidade. Assim, existem processos independentes para tratar de cada tipo de comunicação e *threads*, também independentes, que constroem e gerem as mensagens a circular no sistema. De entre todas as mensagens, as mais importantes são as CAM e DENM que, por exemplo, são construídas, enviadas e recebidas em *threads* diferentes de forma a isolar cada tarefa. As mensagens CAM e DENM foram implementadas de acordo com as normas europeias ITS ETSI e são construidas recorrendo à informação das mensagens recebidas (caso das DENM) mas, principalmente, aos dados fornecidos pelos sensores. Cada sensor é também controlado por um processo independente que serve de interface entre o *software* e os periféricos.

Os processos que gerem os sensores e as mensagens da rede veicular comunicam através de memória partilhada e constituem, em termos de arquitetura, a camada de suporte à aplicação (*Application Support*). Como o nome indica, esta tem a função de fornecer meios para que se possam desenvolver aplicações veiculares, mas, só por si, não constitui uma aplicação veicular.

Assim, no âmbito desta dissertação também se desenvolveu uma aplicação veicular simples que permite o controlo da aquisição de dados dos sensores, envio/receção de mensagens, controlo da comunicação com o *smartphone* e também o uso de uma rede auxiliar para monitorização e atualização da plataforma (Rede Celular). Todas estas funções, exceto a comunicação com a rede auxiliar, foram implementadas tendo por base a camada de suporte à aplicação.

Este capítulo está organizado em dez secções. A primeira aborda os dispositivos e interfaces escolhidos de forma a implementar a arquitetura de *hardware* proposta (secção 5.2), sendo a única secção que se dedica ao *hardware*, todas as outras referem-se ao *software* implementado. A secção 5.3 diz respeito a generalidades e considerações que têm impacto em toda a implementação de *software*. Posteriormente, em Gestão de Sensores (secção 5.4), discute-se o modo de gestão e ligação a cada sensor ligado à unidade de processamento central. A secção 5.5 explica o modo como a informação recolhida é tratada, enquanto que a secção (5.6) fala sobre a forma de construção, envio e receção de mensagens CAM e DENM. A secção 5.7 aborda a integração da rede celular 3G/4G no sistema e a secção 5.8 a implementação da aplicação de segurança. Na secção 5.9 seguinte é apresentado o ambiente de desenvolvimento utilizado. A secção 5.10 aborda a integração do sistema com o módulo de segurança (encriptação de mensagens) e com o módulo WSMP (comunicação com a rede 802.11p). Por fim, é apresentado um breve sumário dos temas abordados neste capítulo – secção 5.11.

5.2 Plataforma de *Hardware*

Neste ponto são abordadas e justificadas as escolhas feitas em termos de *hardware* para a plataforma. Isto com base no que está disponível no mercado, no preço e nas funcionalidades que cada elemento oferece.

A escolha destes componentes também teve em atenção a plataforma de comunicações ITS construída no âmbito do projeto HEADWAY. Em [32] foi definida a estrutura geral da plataforma *IT2S* (figura 5.1) que apresenta um interface de comunicação com o restante *hardware* baseado em USB. Nesse documento também se pode verificar que a principal forma de interação com o utilizador HMI é através de um *smartphone* que utiliza o mesmo interface.

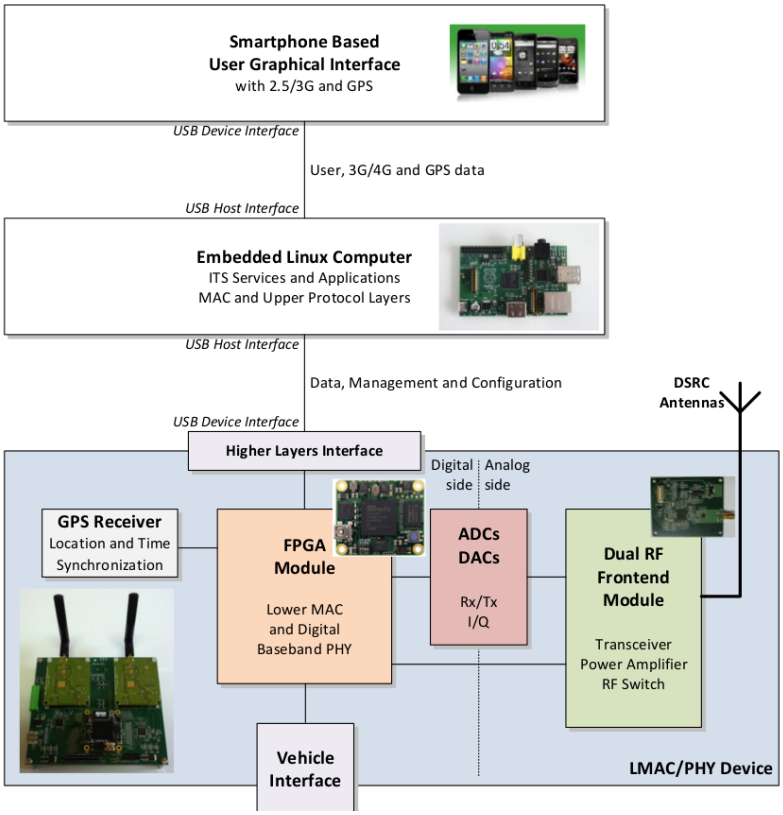


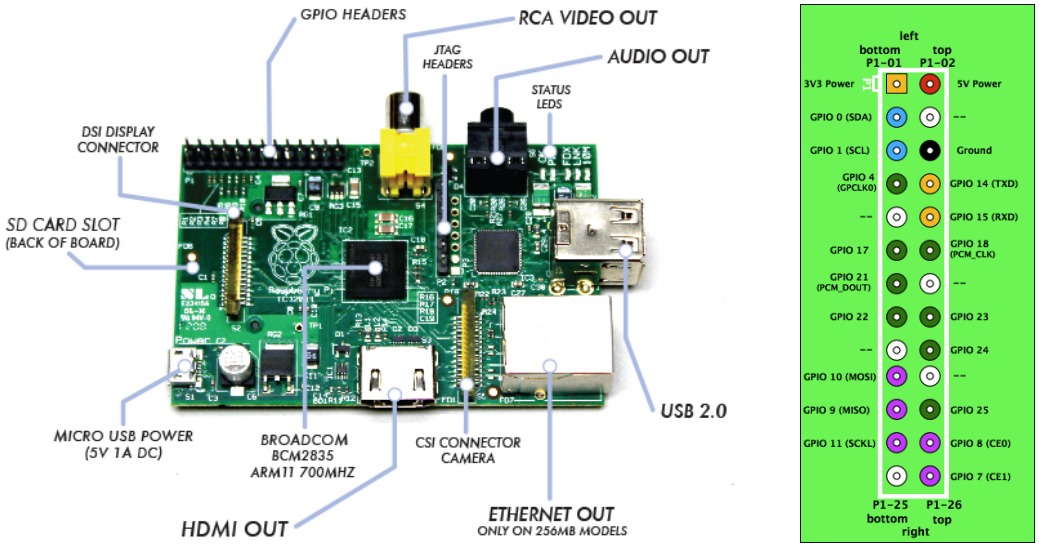
Figura 5.1: Plataforma HEADWAY IT2S (*fonte:* [32])

5.2.1 *Single Board Computer*

O nível de complexidade da aplicação levou a que fosse escolhido um SBC com um processador de arquitetura ARM e SO Linux como base (unidade de processamento cen-

tral). Isto permite não só uma maior flexibilidade e rapidez no desenvolvimento do software como uma maior capacidade de processamento quando comparado, por exemplo, com micro-controladores comuns. Outra característica relevante destes dispositivos é a facilidade na ligação a periféricos, o que é uma mais-valia pois existe a necessidade de se usar vários sensores.

De modo a escolher o dispositivo que melhor se adapta ao este propósito, em Setembro de 2012, foi efetuada uma pesquisa por dispositivos que cumprissem os requisitos descritos em 2.5. Desta pesquisa resultou uma tabela comparativa das características dos vários SBCs disponíveis no mercado e o dispositivo que mais se ajustava era o SBC da Gumstix, o *Overo IronStorm COM* com a placa de expansão *Summit* [33]. Este dispositivo apresenta características de robustez, fiabilidade, suporte e flexibilidade que o colocam à frente da concorrência quando se pretende elaborar uma OBU/ RSU para o mercado. Mas, como se pretende implementar um protótipo e não o produto final, escolheu-se um outro SBC, com a mesma arquitetura (ARM), que apresenta boas características, que é substancialmente mais barato, e que para o propósito de desenvolvimento de um protótipo é o ideal – o Raspberry Pi Model B [34]. Além disso, o *software* que for desenvolvido para este dispositivo poderá ser facilmente portado para o anterior.



(a) Raspberry Pi Model B (*fonte:*[35])

(b) Detalhe GPIO (*fonte:*[36])

Figura 5.2: Raspberry Pi Model B

O Raspberry Pi Model B é um dispositivo que apresenta as seguintes caraterísticas [37]:

- SoC Broadcom BCM2835 (CPU, GPU, DSP, and SDRAM);
- CPU: 700 MHz ARM1176JZF-S core (ARM11 family);
- GPU: Broadcom VideoCore IV,OpenGL ES 2.0,OpenVG 1080p30 H.264 high-profile encode/decode;
- Memory (SDRAM): 512 Megabytes (MiB);
- Video outputs: Composite RCA, HDMI, DSI;
- Audio outputs: 3.5 mm jack, HDMI;
- Onboard storage: SD, MMC, SDIO card slot;
- 10/100 Ethernet RJ45 onboard network;
- Storage via SD/ MMC/ SDIO card slot;
- CSI Camera Interface;
- GPIO (*general purpose I/O*).

Este mini computador quando comparado com os computadores modernos tem um desempenho mais baixo, mas uma análise descrita no site da Engadget refere que se consegue reproduzir vídeo de alta definição com áudio sem problemas [38]. Assim, para as aplicações veiculares mais comuns (de segurança, por exemplo), este SBC não deverá ter problemas de desempenho. Além disso, o Raspberry Pi apresenta um conjunto de pinos que permitem uma grande liberdade de escolha dos interfaces com periféricos. Estes pinos podem ser configurados para uso geral (*General Purpose Input/Output* (GPIO)), mas também para uso de protocolos de comunicação mais específicos como *Serial Peripheral Interface* (SPI), *Inter-Integrated Circuit* (I2C) e *Universal Asynchronous Receiver/Transmitter* (UART) [34][36]. Estes passos e opções de configuração podem ser consultados no *datasheet* da BroadCom [34].

Relativamente ao Sistema Operativo (SO), é recomendado o uso do *Raspbian*, uma versão do Debian Linux adaptada ao Raspberry Pi. No entanto, pode utilizar-se Fedora ou, se se pretender um SO Linux só com as funcionalidades e os pacotes de software estritamente necessários, Arch Linux [39].

Do ponto de vista de ligações/interfaces exteriores o Raspberry Pi pode ser representado da forma que se pode ver na figura 5.3. Além dos interfaces mais comuns temos também, através da ligação GPIO, outros interfaces como SPI, I2C e UART.

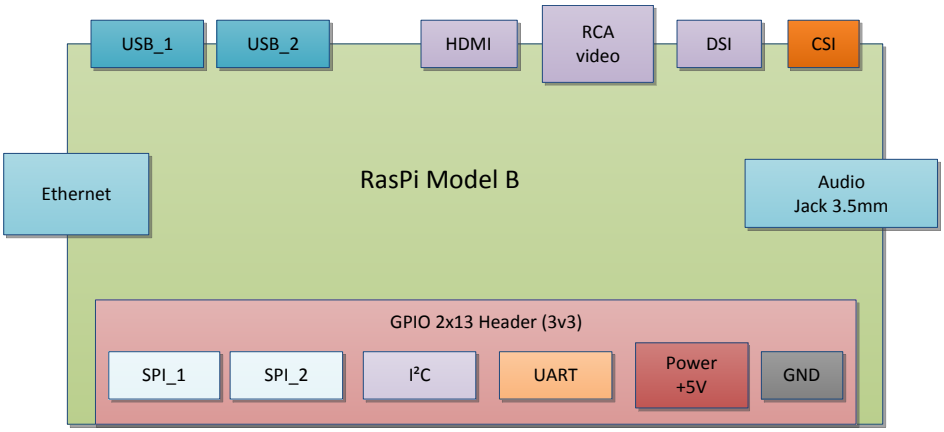


Figura 5.3: Interfaces do Raspberry Pi Model B (baseado em [37])

Assim, existe a possibilidade de ligar os periféricos que são necessários a uma OBU / RSU a esses interfaces. O interface *Ethernet* deverá, em princípio, ser usado como porta de monitorização, *debug* e transferência de ficheiros entre o Raspberry e o Computador de desenvolvimento.

5.2.2 Periféricos/Componentes de *Hardware* Auxiliares

De forma a cumprir com as funcionalidades descritas em 4.2, houve a necessidade de "munir"o Raspberry Pi de outros componentes/ periféricos.

Como foi referido, este SBC dispõe de vários interfaces de ligação, entre eles o USB standard. É óbvio que num produto final e de acordo com o ponto 2.5, não é aconselhável usar este tipo de ligações físicas que são pouco resistentes a vibrações e choques, mas, uma vez que se pretende desenvolver um protótipo e, mais uma vez, por questões de simplicidade, flexibilidade e compatibilidade, optou-se pelo interface USB. Outra razão desta escolha prende-se o facto da interação com o módulo de comunicações veiculares IT2S do projeto HEADWAY e com o *Smartphone* ser feita através de USB [32]. Os periféricos que deverão estar presentes no protótipo final são:

- **Interface OBD-II** - Leitura dos sensores do veículo;
- **Módulo GPS** - Fornece dados de posicionamento através do sistema GPS;
- **Módulo Celular** - Ligação à rede celular.

A lista final de módulos USB compatíveis e testados com o Raspberry Pi pode ser consul-

tada no site de suporte ao mesmo, esta lista é extensa e permite alguma liberdade na escolha de cada um [40].

Um facto que se teve em conta é o de que cada porta USB pode apenas fornecer 140 mA de corrente elétrica e, por exemplo, um smartphone moderno, quando ligado através de USB, consome cerca de 1000 mA. Deste modo, optou-se por introduzir no sistema um *Hub* (concentrador) USB alimentado para que não hajam problemas de falta de corrente ou sobrecarga dos circuitos de alimentação do SBC.

5.2.2.1 Módulo GPS

Existem vários recetores GPS USB que são compatíveis com o Raspberry Pi e com o SO Linux. Entre eles temos:

- GlobalSat BU-353;
- BlueNEXT BN-903GPS;
- HOLUX M-215.

A lista completa e atualizada pode ser encontrada em [40].

Estes módulos são reconhecidos pelo SO Linux através de um driver nativo do *kernel* ou através de um *daemon* chamado GPSD que lida diretamente com o dispositivo GPS e expõe ao programador uma interface comum de comunicação com o recetor [41]. Assim, qualquer um deles pode ser escolhido, uma vez que nesta fase de protótipo a preocupação principal não é a extrema precisão dos dados de GPS, mas sim ter uma fonte de dados de posicionamento para um primeira abordagem ao sistema de cooperação entre veículos.

Resta dizer que a frequência típica a que os recetores GPS comuns disponibilizam novos dados é de 5Hz.

5.2.2.2 Módulo OBD-II

Neste ponto, depois de uma pesquisa no mercado, verificou-se que os dispositivos mais capazes são os baseados no chip ELM327 da Elm Electronics [42]. Este chip é a base de muitos conversores USB - OBD-II, suportando vários protocolos de comunicação presentes nesse interface [43]:

- SAE J1850 PWN;
- SAE J1850 VPW;

- ISO 9141-2;
- ISO 14230-4 (KWP2000);
- ISO 14230-4 (KWP2000);
- ISO 15765-4 (CAN 250/500 kbps, 11/29 bit).

Um exemplo deste tipo de dispositivos é o ScanTool.net ELM SCAN5 USB [44]. Este dispositivo suporta os protocolos descritos em cima e fornece um interface de programação/leitura de dados uniforme a todos eles, nomeadamente, permite que, num sistema Linux, o interface OBD-II seja tratado como uma simples porta série para onde se enviam comandos de leitura e se recebe respostas com o valor de cada sensor (*Parameter ID* (PID)) [43].

Além disso, a empresa responsável pelo ScanTool.net ELM SCAN5 USB disponibiliza uma ferramenta *open-source* para leitura de vários sensores do carro, facilitando o desenvolvimento e teste de novo software [45].

5.2.2.3 Ligação Rede Celular

Em relação ligação à rede celular existem vários módulos (*dongles*) USB disponíveis no mercado e compatíveis com o Raspberry Pi, entre eles os mais comuns fornecidos pelas operadoras. Alguns dos que são apresentados na lista de hardware compatível são os seguintes [40]:

- Huawei E173;
- Huawei E220;
- Huawei E160;
- Digicom Internet Key 7.2 HSUPA MU372-L01;

Do ponto de vista do sistema operativo esta ligação é tratada como um comum interface de rede. A maior parte dos sistemas Linux modernos reconhecem imediatamente o dispositivo como uma ligação de rede celular.

5.2.2.4 Estrutura Final da Plataforma de Hardware

Na figura 5.4 está representada a estrutura final da plataforma de hardware que serviu de base ao sistema.

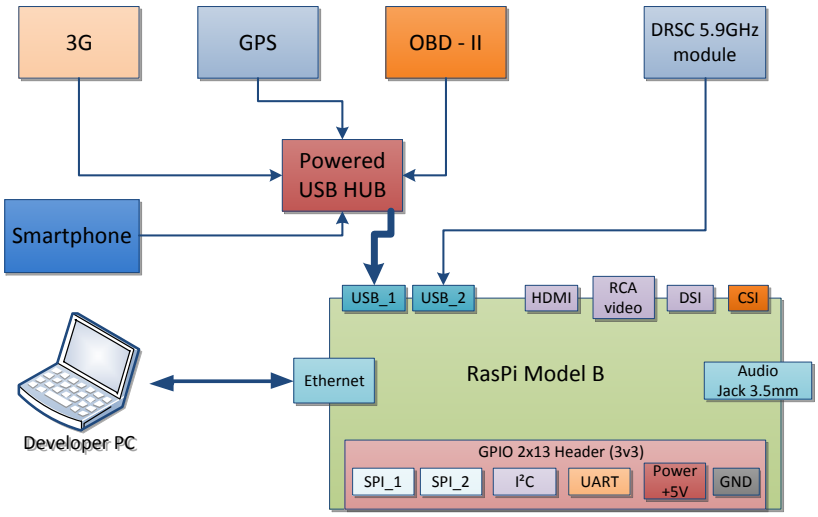


Figura 5.4: Esquema de ligações final do protótipo de OBU/RSU

5.3 Implementação *Software*: Generalidades

Neste ponto irão ser abordadas e justificadas decisões gerais que afetam toda a implementação de *software* que foi elaborada: processos, *threads*, comunicação entre eles e tecnologia por detrás do interface que a camada de suporte expõe à camada de aplicação ISA. A figura 5.5 mostra um esquema da dessa implementação, o qual irá ser explicado em pormenor, em seguida.

Foi definido que cada gestor descrito no esquema da figura 5.5 é um processo independente. A principal razão prende-se com o facto da leitura da maioria dos sensores, comunicação com o módulo de segurança e comunicação com o módulo veicular ser feita de forma bloqueante. Separando o *software* em processos independentes ultrapassa-se essa dificuldade, ao mesmo tempo que se separam e isolam de funcionalidades e dados. Além disso, permite que cada processo se possa facilmente dividir em *threads* se isso se mostrar necessário.

Em relação à comunicação entre processos, esta foi implementada usando uma configuração de memória partilhada. Aqui optou-se por implementar a analogia produtor-consumidor através da utilização de vários segmentos de memória independentes: cada processo (produtor) escreve os dados que pretende partilhar no seu segmento e, após isso, outros processos (consumidores) acedem a esses dados. Deste modo, existe apenas um processo que escreve nessa parte da memória. Apesar desse facto, todos os segmentos de memória partilhados – Controlo e Dados (fig. 5.5) – estão protegidos contra corrupção de dados e condições

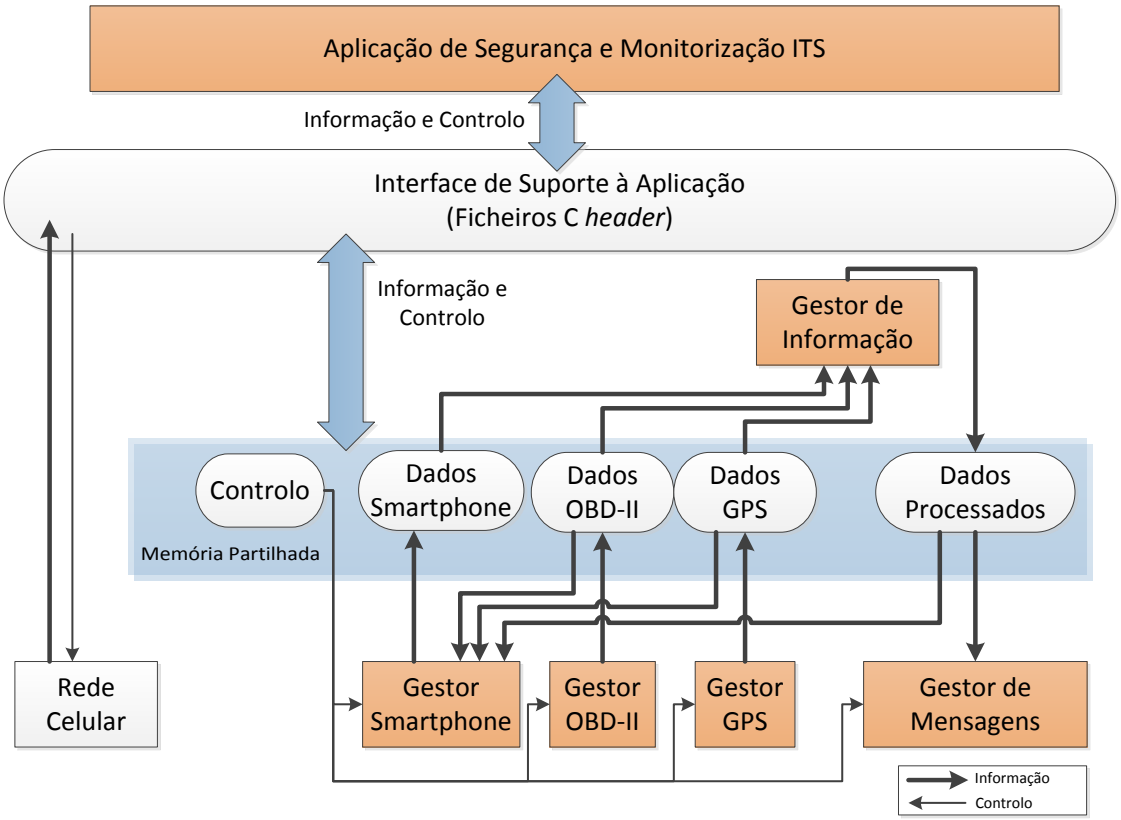


Figura 5.5: Relação e comunicação entre processos

de corrida, uma vez que foram implementados semáforos *Portable Operating System Interface* (POSIX) de forma a garantir exclusão mútua no acesso.

Ainda sobre a comunicação entre processos e a nível da programação que foi feita, cada processo produtor de informação expõe, através de um ficheiro *header*, uma função que permite que outros processos recolham esses dados sem corromper o funcionamento do programa.

Para controlar a execução dos processos existe ainda outro segmento de memória – bloco com o nome Controlo (fig. 5.5) – com variáveis de controlo para cada processo. Consequentemente, cada um deles terá que periodicamente verificar o valor das variáveis respetivas de forma a funcionar do modo pretendido.

O ISA da figura 5.5 corresponde a um conjunto de interfaces em C que permitem que a aplicação aceda a funcionalidades da camada inferior. Esta aplicação é executada num processo separado quem, conceptualmente, representa a camada de mais alto nível da *stack* protocolar.

5.4 Gestão de Sensores

Nesta implementação, existem três processos responsáveis pela gestão e obtenção de dados dos vários sensores: o Gestor do Smartphone, o Gestor OBD-II e o Gestor GPS. À exceção do primeiro que é ligeiramente diferente, estes têm um funcionamento semelhante, ou seja, ligam-se ao dispositivo quando este está presente e começam a recolher dados periodicamente (posição atual, velocidade, aceleração, etc) e disponibilizam-nos no segmento de memória partilhada que lhes corresponde, juntamente com uma marca temporal de modo a garantir a validade dos dados. O Gestor do Smartphone tem ainda a particularidade de poder receber alertas e informações do sistema para que depois sejam enviados ao *smartphone* através da ligação USB.

Nos pontos seguintes, é mostrado, em detalhe, a implementação e funcionamento de cada um.

5.4.1 Gestor OBD-II

Este processo lê os seguintes dados sensores do veículo através do interface OBD-II:

- Velocidade;
- Temperatura do motor;
- Rotações por minuto;
- Indicação de avaria do motor: *Malfunction Indicator Lamp* (MIL).

Como foi utilizado o ElmScan 5, a leitura destes sensores reduz-se ao envio de comandos para porta série do SO, recolher a resposta e converter o resultado para valores decimais, uma vez que a resposta do dispositivo tem um representação hexadecimal (figura 5.6).

Na figura 5.6 é mostrado um exemplo em que o comando "0105"(ler "01"temperatura do motor – PID "05") é enviado obtém-se a resposta como o valor correspondente "7B". Este valor é depois convertido para decimal, sendo que estas conversões de valores, unidades que eles representam e PIDs de cada sensor tiveram por base o código fonte da aplicação *SanTool* que está disponível *on-line* [45].

Assim, tendo em conta este processo, o processo foi implementado do modo representado no diagrama da figura 5.7. Note-se que a aceleração é determinada indiretamente interpolando linearmente, em cada ciclo, dez valores de velocidade recolhidos e em seguida derivando essa função (utilizou-se a biblioteca GSL (*Gnu Scientific Library*)). Houve esta necessidade pois

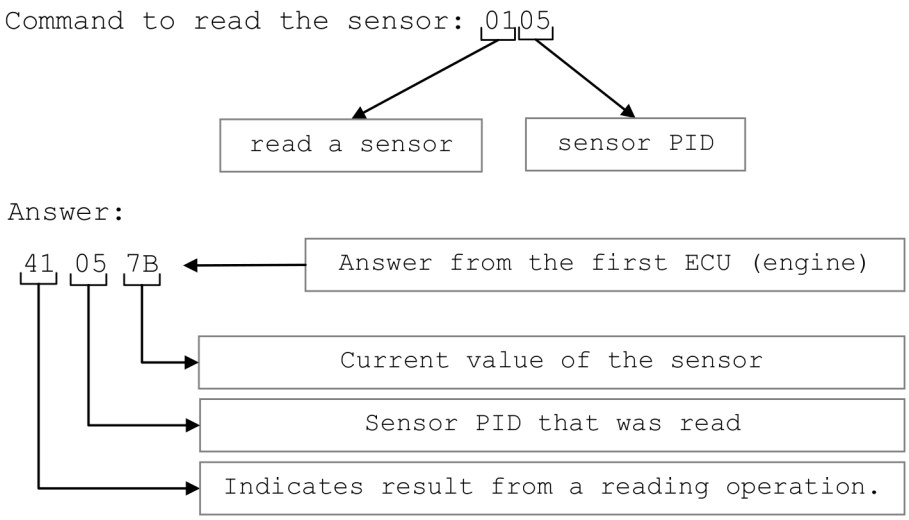


Figura 5.6: Exemplo de leitura da temperatura do Motor.

o acesso ao PID do sensor da aceleração e outros sensores que seriam importantes implica a aquisição de licenças bastantes dispendiosas. Além de que o PID, para o mesmo sensor, varia entre fabricantes.

5.4.2 Gestor GPS

O Gestor de GPS tem a função de recolher os dados sobre o posicionamento e dinâmica do veículo, Nomeadamente:

- Latitude;
- Longitude;
- Altitude;
- Elevação;
- Direção do Movimento (*heading*);
- Velocidade.

Como foi referido nas especificações – ponto 4.3.1 – este processo recorre à plataforma de *software* GPSd, o que permite o suporte a um grande número de recetores GPS no mercado. Em termos de funcionamento é muito semelhante as anterior, como se pode verificar na figura 5.8.

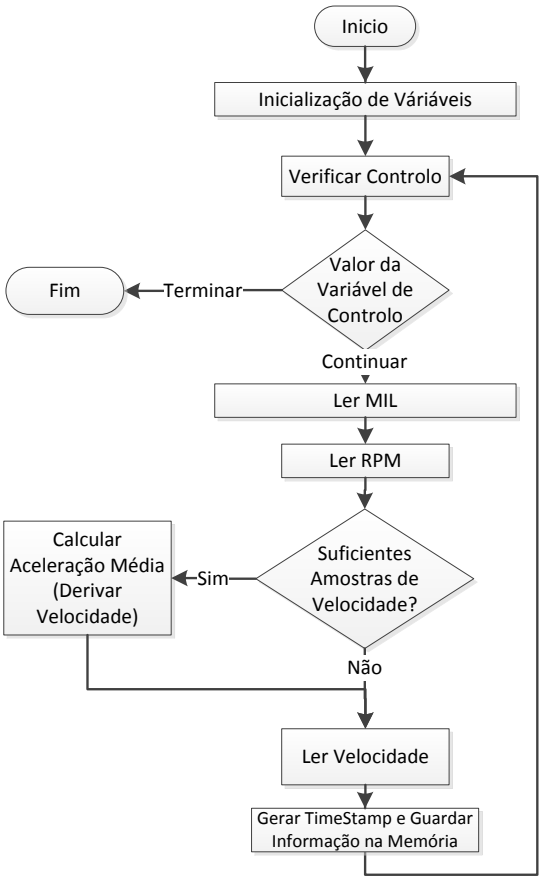


Figura 5.7: Diagrama simplificado do funcionamento do Gestor OBD-II.

5.4.3 Gestor Smartphone

Este processo é responsável pela troca de informação com o *smartphone Android*. Mais precisamente, recebe do *smartphone* o seguinte:

- Alertas sobre acidentes detetados - estes alertas contêm o tipo e a gravidade do acidente, bem como a sua localização.
- Aceleração lida pelo acelerómetro.

E é enviado para o *smartphone*:

- Estado de avaria do veículo: com avaria ou sem avaria (MIL);
- Alerta de perigo detetado pelo sistema ou através de uma mensagem recebida pela rede veicular - contém o tipo de incidente/perigo e a sua localização;

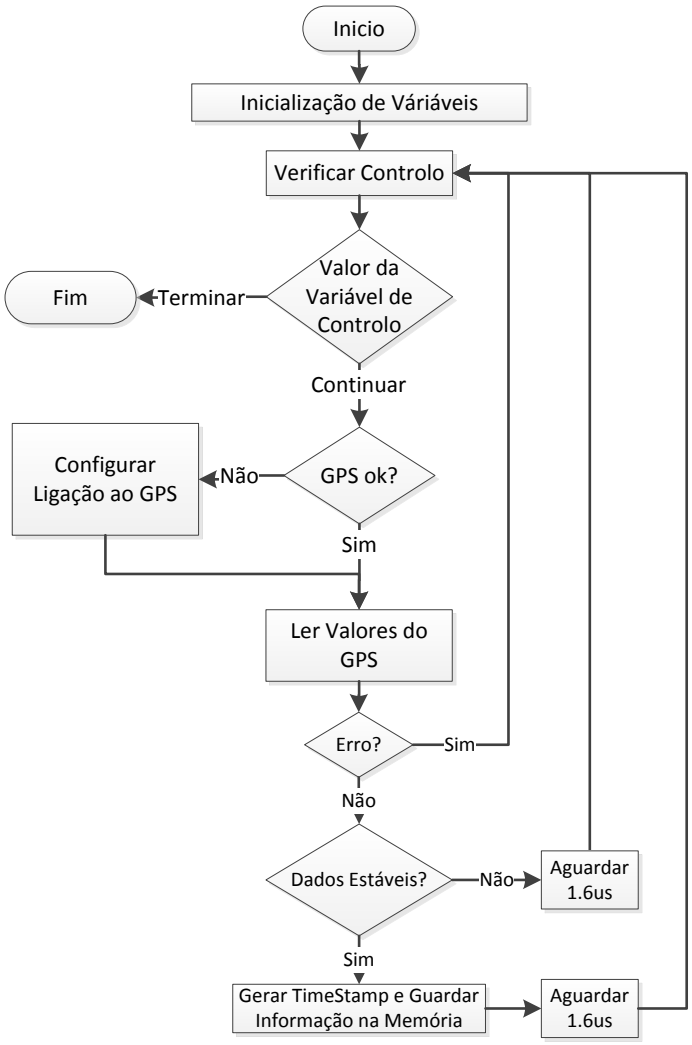


Figura 5.8: Diagrama simplificado do funcionamento do Gestor GPS.

- Velocidade;
- Rotações por minuto do motor;
- Temperatura do motor;

Em termos de funcionamento, a recolha informação dos sensores do *smartphone* é semelhante aos processos que se foram descritos anteriormente: lê os dados e disponibiliza-os no seu segmento de memória. O que o diferencia este processo é a capacidade de obter os dados dos restantes processos e enviá-los para o *smartphone* – figura 5.9.

Uma vez que oferece um acesso simplificado aos dispositivos USB, foi utilizada a biblioteca *open source libusb-1.0* versão 1.0.9 na implementação deste mecanismo de comunicação com

o *smartphone*.

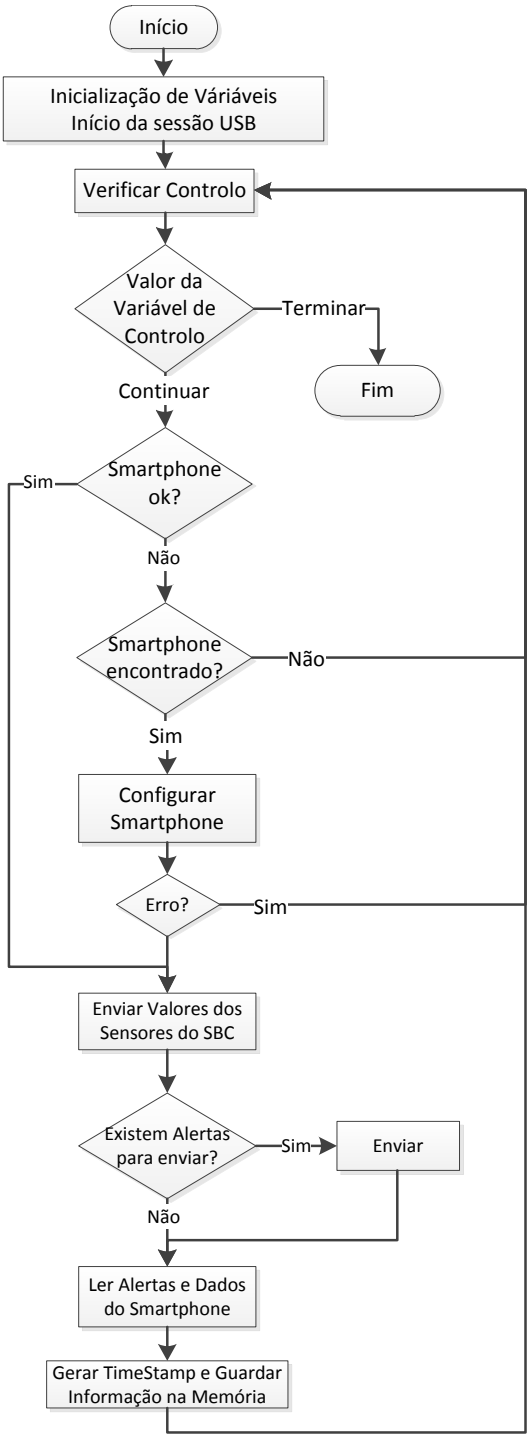


Figura 5.9: Diagrama simplificado do funcionamento do Gestor Smartphone.

Usando o *smartphone* acrescenta-se ao sistema uma maior capacidade de deteção de acidentes, bem como oferece ao utilizador um HMI coerente com as restantes aplicações Android que usa.

5.5 Gestor de Informação

O processamento da informação recolhida com vista a melhorar a fiabilidade da mesma é um aspeto importante no contexto ITS, é deste modo que se chega a conclusões sobre a ocorrência de acidentes, existência de obstáculos, etc.

O Gestor de Informação é o responsável por este processamento: cruzar informação e disponibilizar os dados mais fiáveis e mais coincidentes com a realidade em redor do veículo. Este processo recolhe informação acedendo aos dados da memória partilhada de cada sensor, este funcionamento é retratado no diagrama da figura 5.10.

Em relação ao processamento de dados, este é relativamente simples uma vez que se destina principalmente a "alimentar"o gestor de mensagens cooperativas (5.6), para isso inicialmente existe uma escolha da fonte de informação, por exemplo, é preferida a leitura da velocidade feita pelo interface OBD-II em relação ao GPS, mas quando os sensores do veículo não estão disponíveis o Gestor de Informação, ao ter um número de valores suficientes, calcula a aceleração interpolando os valores da velocidade recolhidos pelo GPS e em seguida derivando essa função, tal como acontece no Gestor OBD-II (5.4.1). Em relação à deteção de avarias é lido o estado da MIL e disponibilizado ao sistema.

Por fim, no segmento de memória a que este processo diz respeito existem ainda outros valores estáticos e que têm de ser definidos pela a aplicação da camada superior. Entre eles o número que identifica a estação (*StationId*), o seu tipo e dimensões.

5.6 Gestão de Mensagens da Rede Veicular

Este processo é responsável pela comunicação com a rede veicular – troca de mensagens CAM e DENM. A sua estrutura aproveita a modularidade da arquitetura descrita em 5.3 que deu a possibilidade de subdividir o processo Gestor de Mensagens em outros fios de execução (*threads*) cada um com funções distintas:

- ***Thread* CAM** - Periodicamente verifica os dados disponibilizados pelo gestor de informação, gera as mensagens CAM no *timming* e nas condições definidas pelo *standard* e passa-as à *thread* de comunicação com a rede veicular;

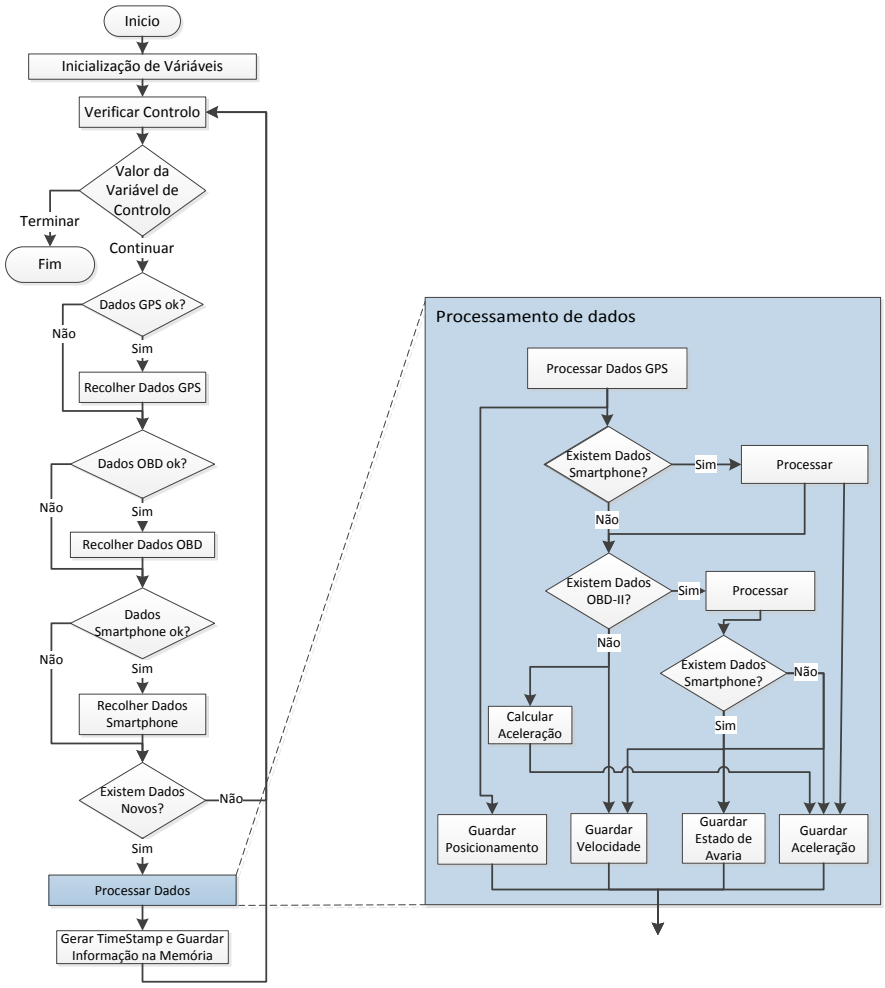


Figura 5.10: Diagrama simplificado do funcionamento do Gestor da Informação.

- ***Thread* DENM** - Periodicamente verifica os dados disponibilizados pelo gestor de informação e os alertas vindos do *smartphone*, gera as mensagens DENM no *timming* e nas condições definidas pelo *standard* e passa-as à *thread* de comunicação com a rede veicular;
- ***Thread* Comunicação Veicular** - Esta *thread* encripta e passa as mensagens internas à camada inferior (de rede). Por outro lado, recebe as mensagens da camada de rede, desencripta-as/verifica a sua validade e passa-as à camada de aplicação para que sejam processadas/interpretadas.

Com esta divisão, a troca de mensagens é separada da construção das mensagens, bem como a construção de cada tipo de mensagem é independente. Optou-se por esta estrutura

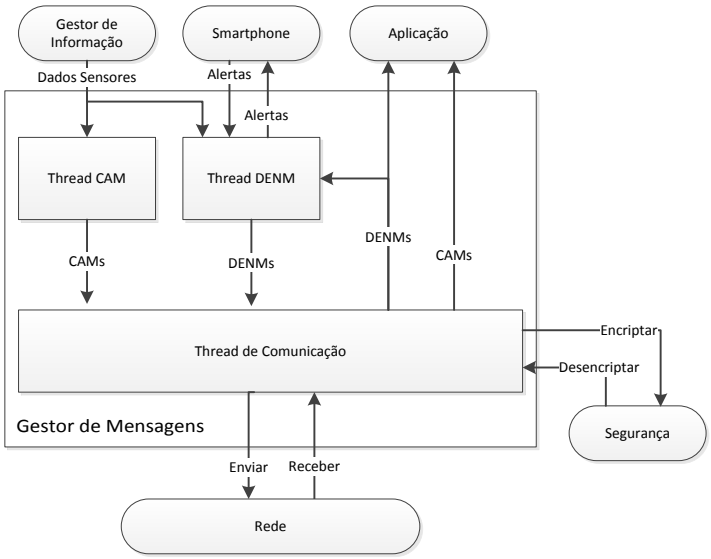


Figura 5.11: Estrutura do Gestor de Mensagens

(figura 5.6) para evitar atrasos no envio das mensagens. A passagem de mensagens das *threads* que geram as mensagens e para a *thread* que as envia para a camada de rede é feita através de filas (POSIX *Queues*) pois, tratando-se de passagem mensagens, este é o mecanismo mais adequado.

Nos pontos seguintes, será explicado detalhadamente a construção das mensagens e o funcionamento de cada *thread*.

5.6.1 Compilador ASN.1

Como foi referido em 3.2, nos standards da ETSI a estrutura das CAM e das DENM encontra-se definida numa linguagem descritiva chamada de ASN.1. Mas se poder trabalhar com essas mensagens é necessário traduzir-las desse formato para estruturas de dados reconhecidas pela linguagem de programação que se vai utilizar – linguagem C.

Claro que esta tradução poderia ser feita manualmente, mas isso levaria muito tempo. Deste modo utilizou-se um compilador que recebe as estruturas de dados definidas no formato ASN.1 e os traduz para linguagem C – o ASN1C [46]. Este compilador é *open-source* e apresenta funcionalidades que são indispensáveis para o uso destas mensagens como a codificação das estruturas de dados segundo a norma UPER.

A versão do compilador ASN1C utilizada foi a v0.9.24.

5.6.2 *Thread* CAM

De forma a cumprir o standard (3.2.1), esta *thread* monitoriza os dados que disponibilizados pelo Gestor de Informação e inicia a construção de uma nova mensagem CAM quando:

- A direção do movimento alterou-se em mais de 4 graus;
- A velocidade do veículo alterou-se em pelo menos 1 metro por segundo;
- A posição alterou-se pelo menos 5 metros;
- Decorreu mais de 1 segundo desde a última mensagem CAM enviada.

Por outro lado, apesar destas condições, a *thread* não gera novas mensagens se isso implicar uma frequência de envio maior que 10Hz (10 por segundo).

Como referido em 5.6.1, da tradução da estrutura da mensagem CAM do formato ASN.1 para C resultaram estruturas de dados utilizáveis nessa linguagem e um conjunto de funções auxiliares, entre elas, uma função que permite codificar a mensagem em UPER. Utilizando essa função, esta *thread* codifica a mensagem e coloca-a na fila de envio para ser enviada.

5.6.2.1 Construção de Mensagens CAM

Para construir cada mensagem CAM, a *thread* utiliza os dados do gestor de informação, mas observando os campos da mensagem (figura A.1) verificou-se que seria necessário aceder a mais sensores do veículo para preencher todos os campos obrigatórios, como isso não foi possível (5.4.1), optou-se por preencher os mais importantes, ou seja, relacionados com a posição, trajetória e estado do veículo. Os restantes campos obrigatórios são valores fixos que permitem que se teste a implementação de envio e receção de mensagens CAM que se fez.

Os campos preenchidos com dados reais são:

- Estrutura `ItsPduHeader`;
- `stationID`;
- Estrutura `stationCharacteristics`;
- `ReferencePosition`: `*heading`, `longitude`, `latitude`, `elevation`;
- `vehicleCommonParameters`: `vehicleType`, `stationLength`, `stationWidth`, `vehicleSpeed`, `longAcceleration`, `crashStatus`.

As primitivas de construção das mensagens foram implementadas de forma a que o preenchimento dos restantes campos seja fácil, sem alterações de maior no código. Além disso,

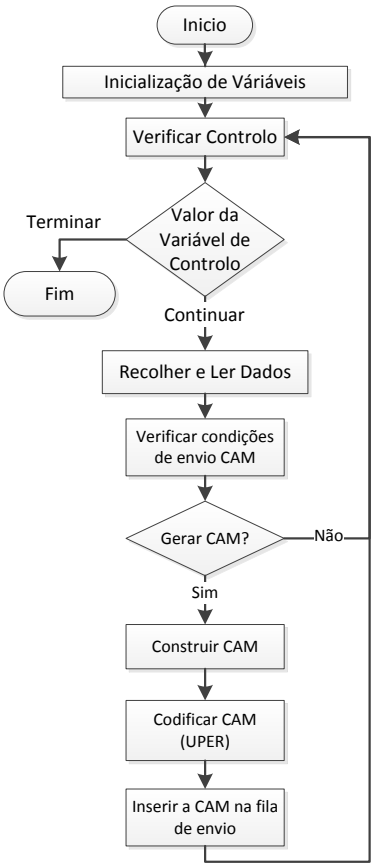


Figura 5.12: Diagrama simplificado de funcionamento da *Thread* CAM

estas primitivas suportam também os campos opcionais. O preenchimento de todos estes campos poderá fazer parte de um trabalho futuro (ponto 7.2).

5.6.3 *Thread* DENM

Esta *thread* funciona de um modo parecido com a anterior (figura 5.13): Monitoriza a informação disponível no Gestor de Informação (5.6.2), gera e coloca as DENMs na fila de envio para que sejam enviadas. Além disso, recebe alertas do processo gestor do smartphone de forma a gerar uma mensagem DENM para cada um deles. Esta comunicação é uma vez mais efetuada recorrendo a filas (POSIX Queues). Assim, a geração de mensagens DENM é provocada por dois fatores: indicação de mau funcionamento do veículo (MIL) ou pela deteção de algum acidente por parte do *smartphone*. As mensagens referentes à segunda causa terão um grau de severidade dependente do alerta recebido e as primeiras terão o grau de severidade mínimo permitido pelo standard (3.2.2).

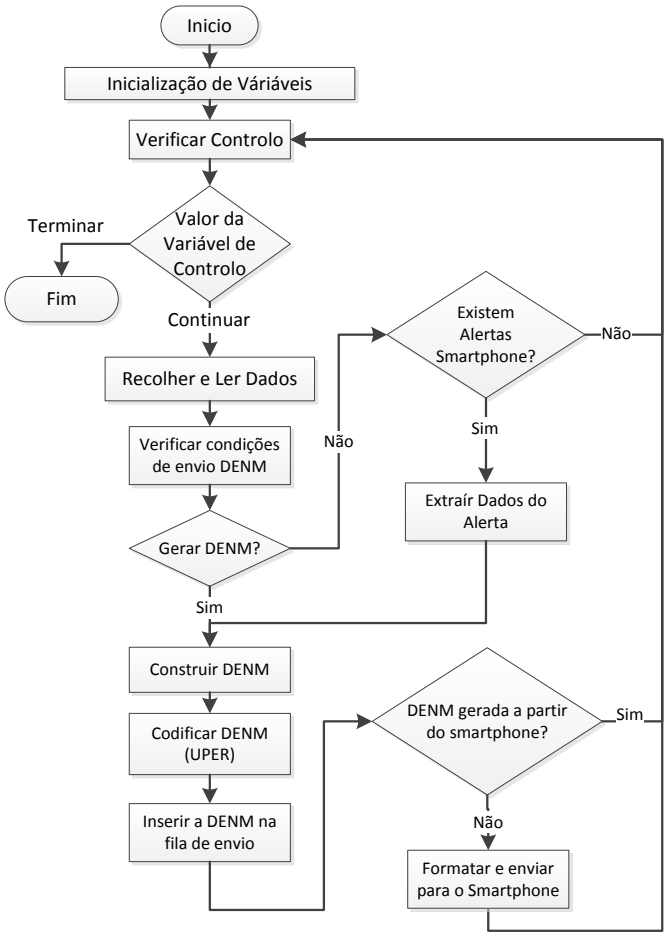


Figura 5.13: Diagrama de funcionamento simplificado da Thread DENM.

Na implementação que foi feita, a DENM gerada é extremamente básica, pois apenas implementa de forma incompleta o *usecase Stationary Vehicle - Accident*. Nesta os valores que são coincidentes com a realidade são:

- *ITS Header*;
- ***Management Container*:** O ID da estação e o número de sequência da mensagem (os restantes têm valor fixo);
- ***Situation Container*:** *Cause Code*, *Subcause Code* e *Severity*;
- ***Location Container*:** Latitude, longitude e altitude do evento ocorrido.

Os dados contidos nos alertas do *smartphone* são extraídos e incorporados na estrutura DENM definida pelo standard de forma a que os condutores dos veículos vizinhos tenham

conhecimento da ocorrência do evento e da sua localização.

5.6.4 *Thread* Comunicação Veicular

Esta *thread* é responsável por enviar e receber para/da camada de rede as mensagens CAM e DENM. Além disso, é responsável por encriptar as mensagens enviadas e desencriptar/validar as recebidas. Para isso, esta *Thread* recorre a aplicações de rede e de segurança desenvolvidas no âmbito da dissertação de outros dois membros da equipa HEADWAY IT. A ligação entre as várias aplicações é abordada no ponto 5.10.

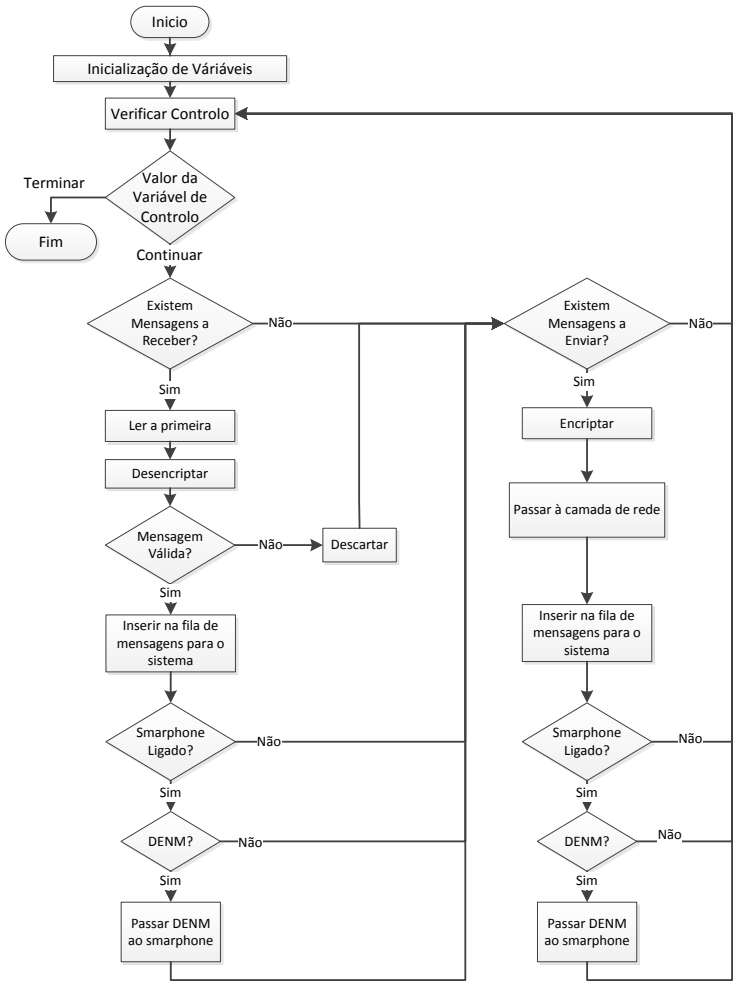


Figura 5.14: Diagrama de funcionamento simplificado da Thread de Comunicação Veicular.

O funcionamento desta *Thread* é descrito no diagrama da figura 5.14 e consiste, basicamente, em monitorizar a existência de mensagens para enviar (fila de envio), encriptar e

passar as mesmas para a camada de rede. Por outro lado, as mensagens recebidas são desencriptadas/validadas e passadas ao sistema (fila de mensagens recebidas) e, existindo um *smartphone* ligado, passar as mensagens DENM ao gestor do smartphone para que o condutor seja alertado. Além das mensagens recebidas, esta *thread* insere na fila de mensagens cooperativa do sistema todas as mensagens enviadas, deste modo uma aplicação veicular poderá monitorizar as mesmas. A distinção entre mensagens recebidas e enviadas é feita, nessa camada, comparando o campo *StationID* da mensagem com o *StationID* da própria estação, assim, simplificou-se o fluxo de mensagens.

5.7 Rede Celular

O bloco "Rede Celular" presente na figura 5.5 representa uma rede extra que o sistema operativo Linux poderá dispor mediante a ligação dispositivo de rede extra ao SBC. Quer seja um dispositivo 3G, 4G, WiMAX ou WiFi e utilizando as *drivers* apropriadas, este deverá estar disponível na forma de um interface de rede comum (eth0, wlan0, etc) que a aplicação e a camada de suporte ITS poderão utilizar para os mais diversos fins. Por exemplo, na implementação da Aplicação Cooperativa de Segurança (5.8), este interface foi utilizado para aceder a uma base dados remota para monitorização e atualizações.

Em conclusão, este bloco representa as funcionalidades do SO que servem também de suporte ao sistema implementado nesta dissertação.

5.8 Aplicação Cooperativa de Segurança ITS

Esta aplicação de segurança usa os recursos da camada de suporte de forma a que o sistema funcione com um todo, iniciando cada recurso da camada inferior. Além disso, dispõe de uma funcionalidade extra que se pretende com o reporte dos dados sobre a própria estação e estações vizinhas para uma base de dados remota, utilizando a rede de *backup* (3G/4G). Esta aplicação é então responsável por (ver figura 5.15):

- Alocação da memória partilha para comunicação entre processos;
- Configuração do sistema a partir do ficheiro *obu_host.conf*;
- Iniciar os vários processos;
- Monitorizar a troca de mensagens no sistema;
- Informar o utilizador do estado atual do sistema;

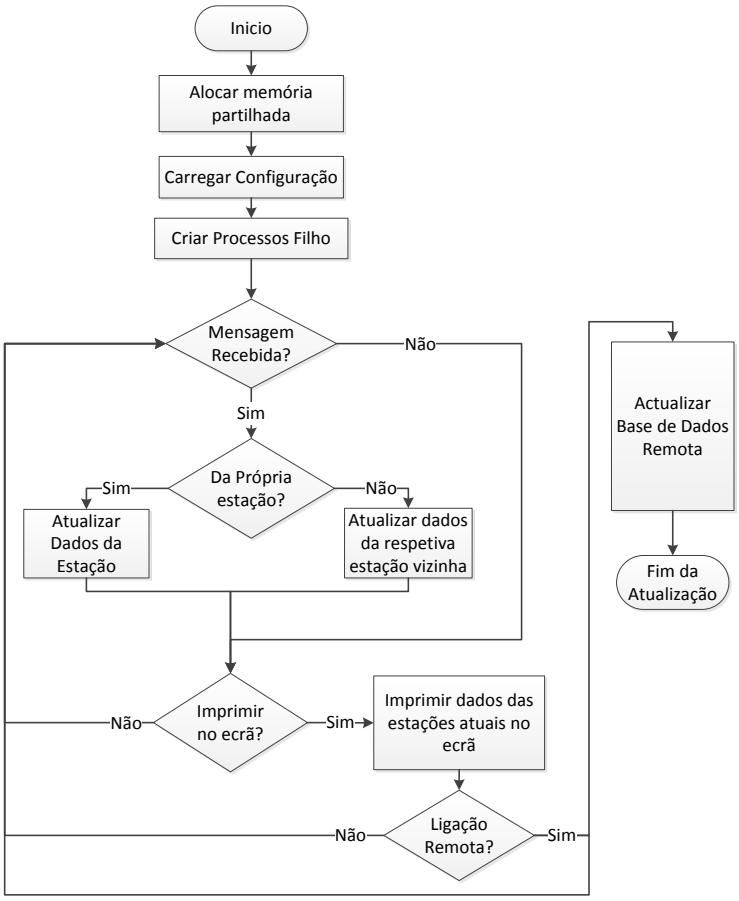


Figura 5.15: Diagrama de funcionamento simplificado da aplicação de segurança veicular.

- Possibilitar a gestão remota das estações;
- Encerrar corretamente o sistema.

A monitorização das mensagens do sistema é feita através da recolha das mesmas de uma fila que contém cada mensagem CAM e DENM criada ou recebida pelo sistema. Estas mensagens são, em seguida, processadas e os dados que contêm são armazenados em memória. Em intervalos regulares de tempo, os dados das estações armazenadas (própria e vizinhas) são mostrados ao utilizador. Existindo ligação a uma rede de *backup*, e caso se pretenda, estes dados são também exportados para uma base de dados remota recorrendo às bibliotecas mySQL disponíveis para Linux.

Esta aplicação tem funções simples, mas permite ao utilizador visualizar e usufruir dos recursos oferecidos pela camada de suporte à aplicação.

5.9 Ambiente de Desenvolvimento

Uma das premissas iniciais desta dissertação é que o sistema tivesse por base o SO Linux por ser de código aberto e, principalmente, altamente personalizável e flexível. Assim, de forma a manter a coerência, o ambiente de desenvolvimento baseou-se num computador de uso comum, arquitetura x86 a correr distribuição Linux "Ubuntu 12.04 LTS"32 bit, com o *kernel* 3.2.0.

Também por questões de flexibilidade e facilidade de integração com outros blocos de software já implementado (Segurança e Camada de Rede), optou-se pela linguagem de programação C. Consequentemente, o compilador escolhido foi o GCC (*Gnu C Compiler*).

O *software* foi desenvolvido e testado no computador de desenvolvimento e depois compilado no SBC Raspberry Pi. Deste modo, não houve a necessidade de programar diretamente sobre o sistema.

5.10 Integração com a Camada de Rede e Camada de Segurança

A camada de suporte à aplicação e a aplicação sem as camadas protocolares adjacentes oferece uma solução ITS incompleta. Então integrou-se o *software* desenvolvido nesta dissertação com um módulo de que adiciona segurança ao sistema e outro que permite a ligação à rede 802.11p. Estes foram desenvolvidos por pessoas diferentes e em separado, assim houve a necessidade de encontrar um método de comunicação entre os módulos que fosse flexível e que não implicasse mudanças profundas no código de cada um.

O módulo de segurança fornece serviços de encriptação de mensagens baseados no standard IEEE 1609.2 e o módulo de rede utiliza a norma IEEE 1609.3, implementando o protocolo WSMP para as trocas de mensagens com a rede, esta troca é efetuada através da plataforma DSRC desenvolvida no Instituto de Telecomunicações de Aveiro – plataforma HEADWAY [32]. Ambos os módulos funcionam em executáveis diferentes e, de forma a poderem comunicar, foram utilizados *sockets* TCP/IP por uma questão de simplicidade e modularidade. A estrutura geral desta comunicação é representada na figura 5.16.

Nesta secção são abordados os detalhes dessa integração que foi motivada pela necessidade de construir um sistema completo: aplicação, segurança e rede.

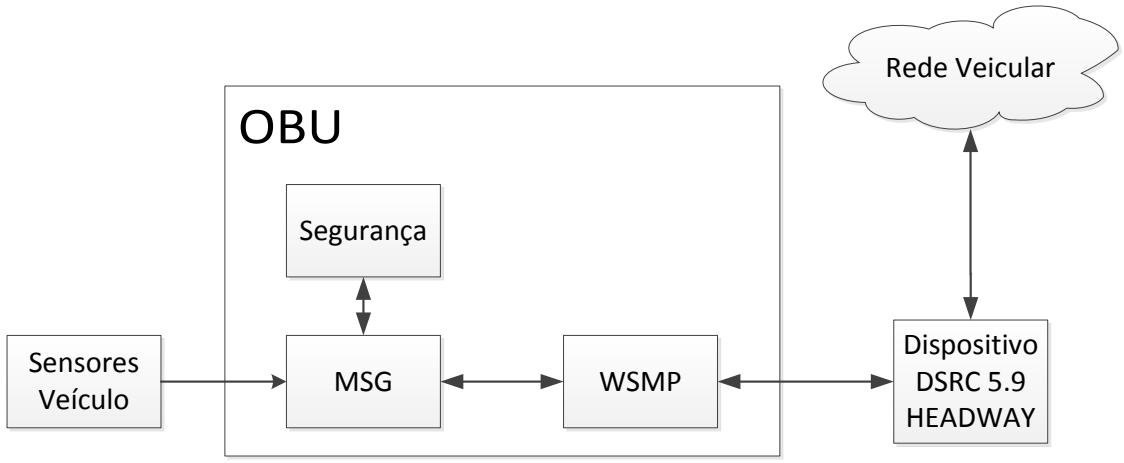


Figura 5.16: Vista Geral da Comunicação entre os vários módulos de *software*.

5.10.1 Módulo de Segurança 1609.2

No módulo de segurança foi implementado, em *software*, o standard *IEEE 1609.2 D17 Draft*. A implementação recorreu à linguagem de programação C e às bibliotecas *Open Secure Sockets Layer (SSL)*, oferecendo os seguintes recursos criptográficos: *Elliptic Curve Digital Signature Algorithm* (ECDSA) 224 e 256, *Elliptic Curve Integrated Encryption Scheme* (ECIES), Algoritmos Hash, geração e verificação de certificados. Em termos de comunicação, este módulo oferece um *socket* para transmissão de informação de forma bidirecional. Este *socket* é bloqueante, quer isto dizer que quando uma aplicação requer a encriptação ou desencriptação de uma mensagem a comunicação fica bloqueada até que haja uma resposta por para do módulo de segurança.

5.10.2 Módulo de Rede WSMP

O módulo WSMP recebe, através de um *socket* não bloqueante, as mensagens das camadas superiores e passa-as para as camadas inferiores dentro do *payload* de um pacote WSMP de acordo com o standard IEEE 1609.3. Ao receber uma mensagem da camada inferior, este módulo realiza a operação inversa, ou seja, recebe o pacote WSMP, extrai a mensagem do *payload* e passa-a à camada superior (ex. aplicação) através do referido *socket*.

Este módulo foi desenvolvido na forma de um *daemon* Linux, isto é, um processo que corre no SO em segundo plano. Para comunicar com ele é necessário que a aplicação se registe para receber e enviar mensagens, utilizando a *Application Programming Interface* (API) fornecida.

A figura 5.17 descreve a estrutura da comunicação com este *daemon*.

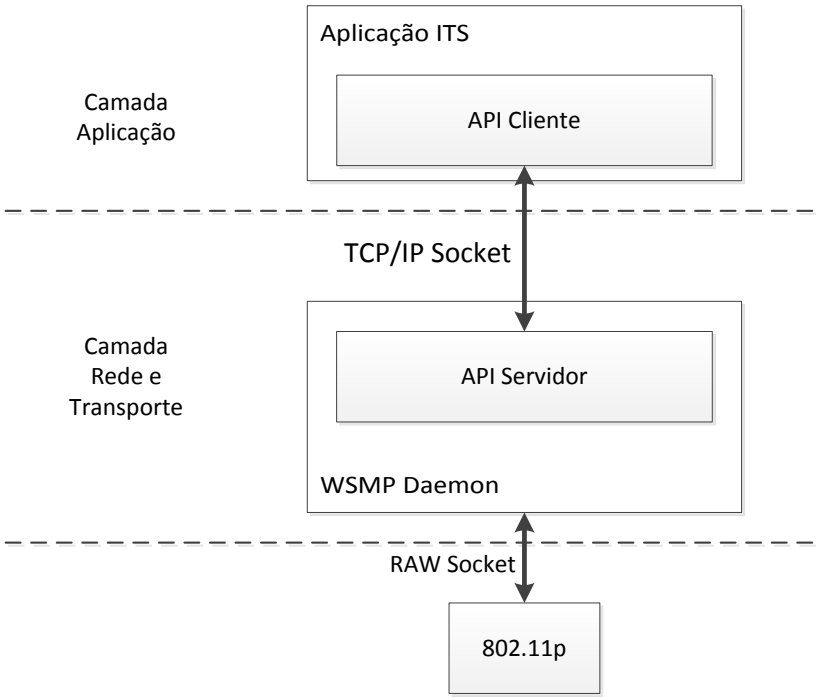


Figura 5.17: Comunicação com módulo WSMP

Resta referir que, além de WSMP, este *daemon* também suporta o protocolo *Fix Adapted for STreaming* (FAST), mas neste caso utilizou-se apenas o primeiro.

5.10.3 Solução de Integração

Depois de compreender o funcionamento e interfaces destes módulos auxiliares foi necessário integrar esses módulos na camada de suporte à aplicação descrita nesta dissertação, mais precisamente na *thread* de comunicação veicular, que é lançada pelo Gestor de Mensagens – ponto 5.6.4.

Em primeiro lugar foi escolhido o procedimento de envio e receção das mensagens. No envio, a *thread* de comunicação veicular passa a mensagem ao módulo de segurança para ser encriptada, acrescentando um byte – carácter 'e' – ao início da mensagem indicando que a mensagem não está encriptada e que se pretende encriptar. Em seguida a mensagem é encriptada enquanto que a *thread* de comunicação fica bloqueada á espera da resposta. Uma vez encriptada, o carácter 'e' é substituído pelo carácter 'd' – indicação de que a mensagem se encontra encriptada – e passada à *thread* que depois a passa ao módulo WSMP para que

seja enviada para a rede. A *thread* de comunicação, ao receber uma mensagem, passa-a imediatamente ao módulo de segurança para que seja desencriptada (quer esteja encriptada ou não). Se o módulo de segurança retornar um erro (mensagem inválida, por exemplo), a mensagem é descartada, caso contrário, a *thread* coloca a mensagem numa fila de saída para que seja processada pelo sistema. Foi necessário a introduzir um *byte* no início da mensagem porque quando na altura de elaboração desta dissertação não existem standards que definam estes interfaces. Este procedimento é esquematizado na figura 5.18.

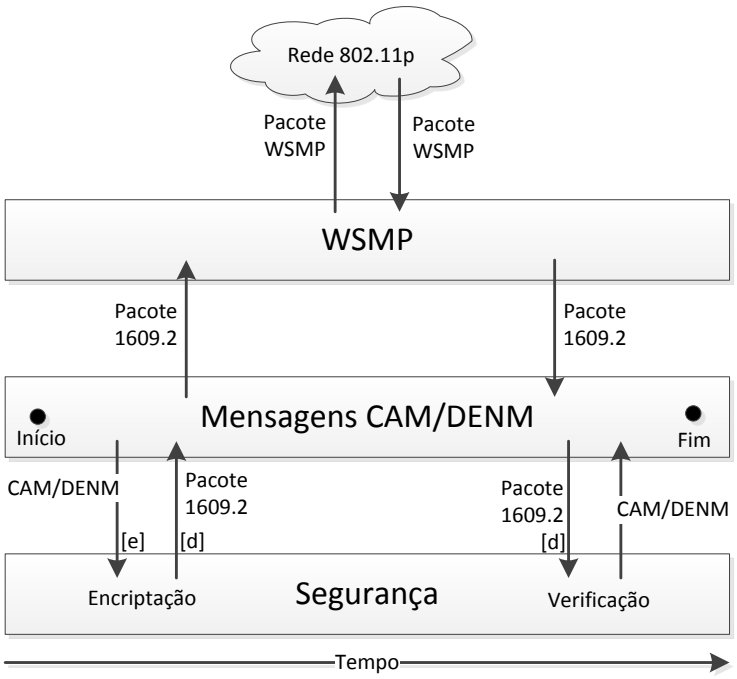


Figura 5.18: Processo de troca de informação entre módulos

É preciso referir que o facto da *thread* de comunicação bloquear à espera de resposta do módulo de segurança não implica a perda de mensagens. Isto porque à entrada desta *thread* existe uma fila que armazena as mensagens a enviar. Por outro lado, o módulo WSMP possui também um fila de saída que guarda as mensagens até que a *thread* as leia.

Com esta integração de módulos obteve-se um sistema completo que recolhe informação, gera, envia e recebe mensagens seguras da rede 802.11p.

5.11 Sumário

Este capítulo ofereceu ao leitor a oportunidade de se inteirar da implementação efetuada no âmbito desta dissertação. Assim formam apresentadas as seguintes implementações:

- Plataforma e interfaces de hardware – Raspberry Pi como unidade de processamento periféricos (recetor GPS, leitor OBD-II, dispositivo 3G/4G e *smarphone*) ligados por USB através de um *hub* alimentado. O módulo **HEAWAY!** (**HEAWAY!**) liga-se também por USB;

- Camada de suporte à aplicação – Processos e *threads* independentes que controlam os vários subsistemas: mensagens da rede, comunicação com sensores e gestão da informação. Estes processos comunicam, maioritariamente, através do método de memória partilhada, mas pontualmente são também utilizadas filas, principalmente, no tratamento das mensagens CAM e DENM;

- Integração – Realizada através de *sockets* que possibilitam a comunicação entre o *software* implementado nesta dissertação, o módulo de segurança IEEE 1609.2 e o módulo de rede WSMP.

- Aplicação – Monitoriza e expõe ao utilizador o estado da própria estação e estações vizinhas. Além disso, armazena esses dados numa base de dados *Structured Query Language* (SQL) remota. Esta ligação remota é efetuada através do dispositivo 3G/4G que oferece um interface de rede e ligação à Internet comum.

Capítulo 6

Avaliação do Sistema

Este capítulo diz respeito à avaliação do sistema, pois, uma vez implementado, é necessário testar a sua validade no termos dos requisitos definidos anteriormente. Esta validação foi feita em laboratório uma vez que a dispositivo DSRC HEADWAY não se encontra devidamente estável para testes deste tipo em ambiente real.

A plataforma de teste usada baseou-se em dois Raspberry Pi que trocam mensagens CAM e DENM entre si. Os dados para construção dessas mensagens – GPS e OBD-II – provêm de simuladores. Mas, para manter a correspondência com o ambiente real, estes são executados num computador (servidor) externo que serve os dois SBCs e evita que lhes seja imputado processamento extra que não existiria usando dispositivos GPS e OBD-II reais. Todos estes elementos comunicam por cabos *ethernet* através de um *switch* de rede comum.

A partir dos ensaios realizados obtiveram-se várias métricas que caracterizam o sistema em termos de cumprimento dos standards ETSI e de outras características temporais relacionadas com a recolha de informação e tratamento da mesma. Através de testes mais simples também foi possível confirmar a validade da implementação da comunicação USB com o *smartphone* Android, bem como a funcionalidade de monitorização da plataforma.

Este capítulo começa com uma pequena introdução, definindo-se logo em seguida as métricas importantes a obter. Para isso começa por explicar-se como compilar e executar o software, ferramentas e arquitetura da plataforma de teste utilizada. Depois são descritos os diferentes cenários utilizados e logo após os resultados obtidos. Por fim, são apresentadas considerações e testes simples efetuados sobre à comunicação USB – Smartphone e à monitorização da plataforma de forma a aferir sobre o seu funcionamento.

6.1 Introdução

Depois do sistema implementado existe a necessidade de o testar e validar. Tendo em conta que se está a implementar um conjunto de standards e que os mesmos impõem regras/requisitos, é de extrema importância garantir que são cumpridos. Olhando para os standards que regem os trabalhos desta dissertação conclui-se que estes requisitos se dividem em fiabilidade e temporais.

Os requisitos de fiabilidade prendem-se com a capacidade do sistema gerar mensagens consistentes com a realidade em volta e de as entregar ao destinatário sem que estas se percam na transmissão. Por outro lado, os requisitos temporais impostos pelo standard (3.2.1.2) têm a ver com o tempo de recolha da informação, tempo de construção, tempo de receção e processamento da mensagem.

Assim, neste capítulo serão avaliados estes requisitos testado o sistema em ambiente de laboratório, uma vez que o seu teste em ambiente real implicaria que a plataforma DSRC HEADWAY [32] estivesse completamente funcional, facto que, nesta altura, não é realidade. Houve também a preocupação de, no desenho e execução dos testes, isolar os vários módulos de *software* (Aplicação/Suporte, Segurança e WSMP) de forma a que os resultados não fossem afetados pelo comportamento dos outros módulos.

6.2 Métricas Utilizadas

Antes de desenhar a forma como os testes irão ser conduzidos, é necessário definir claramente o que se quer medir e testar. As métricas que se pretende obter são as seguintes:

- Duração do processo de recolha de informação;
- Duração da construção, codificação e passagem à camada seguinte de cada mensagem (CAM/DENM);
- Duração do processo de validação e pós-processamento de cada mensagem recebida;

Além disso, pretende-se aferir acerca do funcionamento na comunicação *smartphone* – OBU e da exportação de dados através da funcionalidade de monitorização da plataforma.

Estas métricas terão de ser obtidas, em primeiro lugar, isolando o módulo de *software* elaborado nesta dissertação para que seja possível concluir acerca do impacto que este módulo tem no desempenho do sistema como um todo.

6.3 Compilar e Executar o Software

Antes de testar o *software* existe a necessidade de o saber compilar e, principalmente, como o executar uma vez que este sistema terá que comunicar com mais dois módulos de *software* a correr na mesma máquina em executáveis diferentes (segurança e rede – *ver Integração com a Camada de Rede e Camada de Segurança*). Esta secção aborda, então, os requisitos e procedimentos para a compilação e execução do sistema desenvolvido nesta dissertação.

6.3.1 Requisitos

De modo a compilar com sucesso o *software* é necessário um sistema computacional a correr um SO Linux moderno com as bibliotecas de desenvolvimento básicas (GCC e *C header files*) e também com algumas bibliotecas de desenvolvimento extra:

- *libncurses5* – usada para uma interface de texto melhorada.
- *libusb-1.0* – comunicação com periféricos USB (*Smartphone*, por exemplo).
- *libgps* – utilizada para comunicação com recetores GPS.
- *libgsl0* – (*GNU Scientific Library*) funções matemáticas avançadas;
- *libmysql* – ligação a bases de dados locais e remotas (monitorização e atualização).

Estas bibliotecas dizem apenas respeito ao que foi desenvolvido no âmbito desta dissertação, isto é, um dos três módulos de sistema. O módulo de segurança e o módulo de rede, apesar de não diferirem muito, têm os seus próprios requisitos em termos de *software* de compilação. Além disso assume-se que esses dois módulos já se encontram devidamente compilados para a arquitetura do sistema computacional e que estão prontos a ser executados.

Em termos de *hardware* auxiliar é necessário um recetor GPS que seja reconhecido pela biblioteca *libgps* e, consequentemente, pelo *daemon GPSd*. É também necessário um dispositivo OBD-II baseado no chip *ELM327* para ligação aos sensores do veículo e, por fim, terá de existir um interface de rede para comunicação com outros veículos. Opcionalmente, pode anexar-se ao sistema um *smartphone Android* para HMI e também outro interface de rede com ligação à Internet (*dongle* 3G, por exemplo) para efeitos de monitorização e atualização da plataforma.

Cumpridos estes requisitos, é altura de iniciar o processo de compilação e execução do *software*.

6.3.2 Procedimentos

A compilação e execução deste sistema foram testadas em duas arquiteturas distintas, Intel x86 (PC de uso comum) e ARM11 (Raspberry Pi), o procedimento seguido foi idêntico em cada uma delas. Na figura 6.1 mostra a estrutura de ficheiros de código-fonte do *software*.

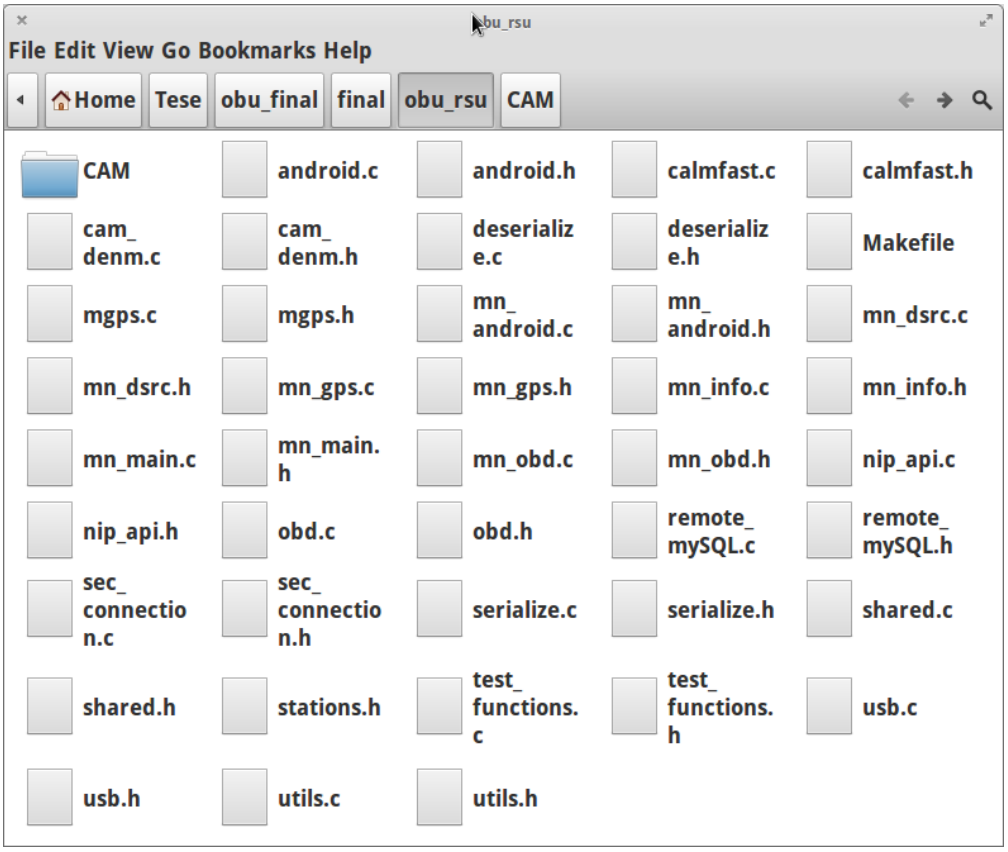


Figura 6.1: Estrutura de ficheiros de código-fonte

Assim, num terminal, começa-se por navegar até pasta "obu_rsu" e depois até à pasta "CAM" e correr o comando *make lib* de forma a que seja criada a biblioteca "libdenm.a". Esta biblioteca contém as estruturas de dados correspondentes à especificação ASN.1 das mensagens CAM e DENM. Em seguida, navega-se até ao diretório superior ("obu_rsu") e executa-se o comando *make*. Isto levará à compilação e *linkagem* do software às bibliotecas auxiliares, inclusive à biblioteca criada anteriormente "*libdenm.a*". Com isto o *software* fica compilado.

Como foi referido no ponto 5.10 (*Integração com a Camada de Rede e Camada de Segurança*) este módulo utiliza mais dois módulos de software – Segurança e Rede (WSMP) – e,

para que funcione corretamente, estes terão que estar ativos antes e durante da sua execução. Antes, também, da execução do programa deve-se navegar até á pasta do mesmo e editar as suas configurações através do ficheiro *"obu_host.conf"*. Nele pode configurar-se o ID da estação, as suas dimensões e o seu tipo, se se pretende usar o dispositivo OBD-II e a porta de comunicação em que ele se encontra e, por fim, existe a possibilidade de se definir se se pretende usar um *smartphone Android* ou não.

Após todos estes passos iniciais, a ordem para colocar o sistema em funcionamento é:

1. Ligar os periféricos: OBD-II e recetor GPS;
2. Num terminal, executar o módulo de rede com o seguinte comando: `sudo ./calmd`;
3. Noutro terminal, executar o módulo de segurança: `./1609dot2`;
4. Ainda noutro terminal na pasta "obu_rsu", executa-se então o programa desenvolvido nesta dissertação: `./HOST_MANAGER`.

Em seguida, o programa deverá começar o seu funcionamento, imprimindo no ecrã os dados relativos ao próprio e aos veículos vizinhos. Além disso, imprime mensagens sobre erros que possam estar a ocorrer.

6.4 Simuladores, Ferramentas e Equipamento Utilizadas

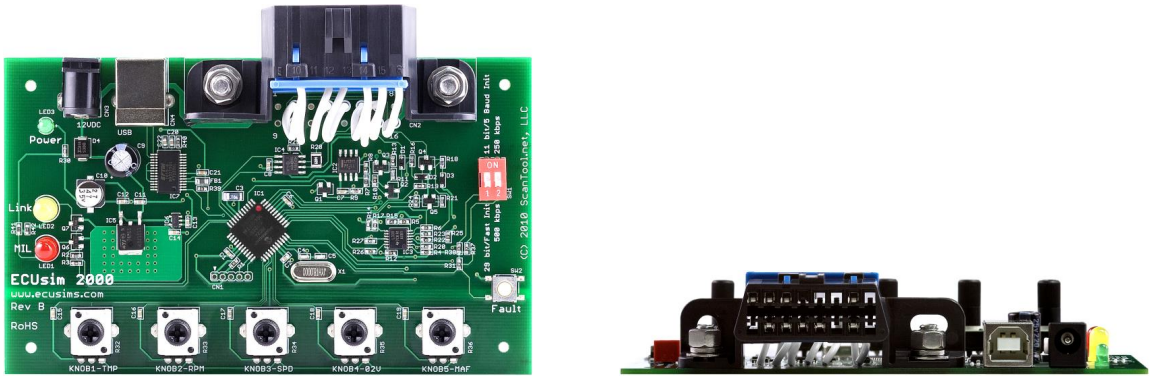
Uma vez implementado foi necessário encontrar formas de testar e validar o sistema. Uma OBU ou RSU opera em ambientes veiculares, a primeira dentro de um veículo e a segunda junto a uma estrada. Mas como se trata de um protótipo inicial, primeiro, deve testar-se o sistema em ambiente laboratorial, para isso recorreu-se a várias ferramentas e simuladores que permitiram que se testasse o sistema em laboratório, tendo sempre em conta o ambiente real.

6.4.1 Simuladores OBD-II

Em primeiro lugar, de forma a testar a comunicação OBU - OBD-II (leitura dos sensores do veículo) foram usados dois tipos de simulador/emulador: um recorrendo a hardware externo – ECUsim 2000 [47] – e outro, por software – OBDsim [48].

O Scantool.net ECUsim 2000 pode simular até quatro *Electronic Control Unit*s (ECUs), expondo, da mesma forma que um veículo moderno um interface OBD-II para que se possa ler o valor dos sensores. Além disso, apresenta cinco potenciómetros a partir dos quais se

pode fazer variar os valores dos sensores mais comuns: velocidade, temperatura do motor, rotações por minuto, nível de oxigénio da mistura e, finalmente, o fluxo de ar que entra no motor. ainda apresenta outro botão que ao ser acionado simula a existência de uma avaria no veículo (figura 6.2). Este simulador suporta os protocolos de comunicação mais comuns na indústria automóvel (*ver* 5.2.2.2). Assim, em conjunto com o leitor Scantool.net ElmScan 5 (*ver* 5.2.2.2) permite que se desenvolva software de leitura dos sensores do veículo em ambiente laboratorial (figura 6.3) [47].



(a) Vista de Cima (b) Vista de Frente

Figura 6.2: Scantool.net ECUsim 2000 (*fonte:* [47])

Figura 6.3: Exemplo de configuração usando o Scantool.net ECUsim 2000 em conjunto com o Scantool.net ElmScan 5

O outro simulador OBD-II é o OBDsim [48] (versão utilizada: 0.16). Este simulador não requer qualquer hardware externo ao sistema operativo, pois trata-se de um software *open-source* que, quando está em execução, apresenta-se ao sistema operativo da mesma forma que o ElmScan 5, isto é, como uma porta série. Para esta porta série (virtual) podem ser enviados comandos da mesma forma que na configuração da figura 6.3. Assim, este software

funciona como um emulador do chip ELM327, respondendo aos comandos no mesmo formato. O OBDsim responde aos pedidos de leitura dos sensores com valores por ele gerados. Essa forma de os gerar pode ser controlada através de argumentos que são passados para o programa [49]. Contudo, após alguns testes simples, verificou-se que o tempo de resposta deste software é bem mais maior do que com o simulador ECUsim 2000, tornando as leituras bem mais lentas.

Assim, recorrendo a este programa, consegue-se desenvolver *software* de leitura OBD-II a partir de emuladores a correr no computador, não havendo, portanto, a necessidade de recorrer a *hardware* externo.

6.4.2 Simuladores GPS

Um inconveniente ao desenvolver *software* para lidar com os recetores de GPS é o facto de, por cada vez que se queira testar uma pequena alteração ou nova função, haver a necessidade de deslocamento para uma zona onde haja sinal e esperar que o dispositivo obtenha os primeiros valores para se poder testar. De modo a ultrapassar este inconveniente usou-se o software GPSFAKE [50].

O GPSFAKE permite, então, que se simule a existência de um dispositivo GPS real ligado ao sistema e que pode ser acedido usando a *framework* GPSd [41]. Para isso, basta passar como argumento ao programa um ficheiro com o registo de um caminho real percorrido anteriormente, depois o GPSFAKE irá reproduzir essa informação ciclicamente.

O GPSFAKE faz parte de conjunto de aplicações que a *framework* GPSd oferece, sendo que a versão utilizada foi a 3.4.

6.5 Arquitetura de Teste

Neste ponto irá ser discutida a arquitetura de teste que foi utilizada e também a forma como se configurou cada o elemento de forma a obter as métricas pretendidas.

6.5.1 Considerações Gerais - *Testbed*

Os dois simuladores utilizados, OBDsim (sensores do veículo) e o GPSFAKE (simulação de GPS), foram desenhados para serem executados na mesma máquina que o *software* que os utiliza, mas isso influenciaria os resultados obtidos uma vez que iria adicionar uma carga de processamento extra que, em ambiente real, não existe. Assim, os testes foram realizados

com base na *testbed* da figura 6.4. Nela a execução dos emuladores OBDsim e GPSFAKE é feita num computador à parte e cada estação (Raspberry Pi) apenas tem de ler os dados que lhe são enviados. Deste modo a carga de processamento em cada estação assemelha-se ao ambiente real. A ferramenta usada como base para esta estrutura cliente-servidor foi o *Socat*, um programa que faz o *forwarding* de portas TCP/*Internet Protocol* (IP) e portas série dentro de uma rede [51].

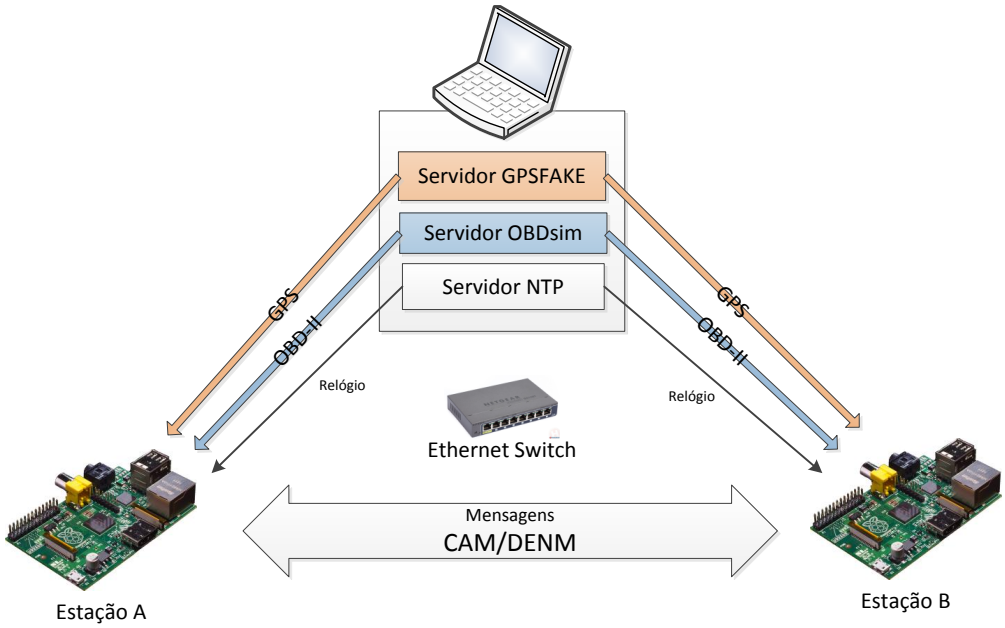


Figura 6.4: *Testbed* usada para os testes.

Em suma, nesta *testbed* todos os elementos comunicam através de uma rede cablada e um *switch*. Dentro da rede, a estação A e a estação B obtêm do computador "servidor"o sinal de relógio de sincronismo (para que as *timestamps* sejam rigorosas e comparáveis), os dados GPS e OBD-II para construírem as mensagens CAM e DENM que trocam entre si.

6.6 Cenários de Teste

Durante esta fase de avaliação obtiveram-se as métricas de 6.2 em cenários diferentes:

- **Cenário 1:** Além do normal envio de mensagens CAM, existe um problema e/ou acidente grave que leva cada uma das estações a também enviar mensagens DENM a uma frequência elevada (≈33Hz) de forma a avisar os veículos vizinhos. Ao mesmo

tempo cada estação também terá de processar e imprimir no ecrã as mensagens que recebe;

- **Cenário 2:** Semelhante ao anterior, mas agora uma mensagem DENM é enviada a cada 2 segundos;
- **Cenário 3:** Modo "normal", isto é, não existe problema nenhum, cada estação apenas tem de enviar mensagens CAM.

Com estes cenários vai ser possível aferir sobre o desempenho de cada estação no envio e receção de mensagens em condições de carga diferentes: carga elevada (cenário 1), carga moderada (cenário 2) e carga ligeira (cenário 3).

6.7 Resultados

Executou-se então o sistema em cada um dos cenários anteriores e foram sendo guardados, em ficheiros e em de cada estação, os tempos em vários pontos do ciclo de execução do programa, permitindo a obtenção de resultados acerca das várias métricas. De referir que houve a necessidade de introduzir em cada mensagem um cabeçalho com 16*bytes* de forma a poderem ser feitas medidas.

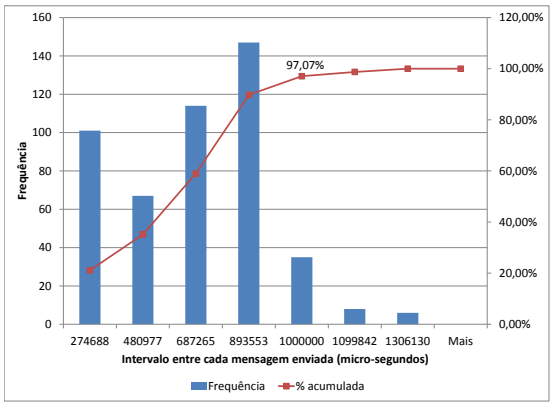
A abordagem destes resultados irá começar pelo que mais interessa, isto é, o cumprimento dos *standards* ETSI para mensagens veiculares em termos de construção, envio e receção de mensagens CAM e DENM. Em seguida, irão ser abordados os tempos de obtenção dos dados e, por fim, são apresentados e discutidos os resultados do sistema como um todo, ou seja, decorrentes da integração de módulos de *software* efetuada.

6.7.1 Envio e Receção de Mensagens

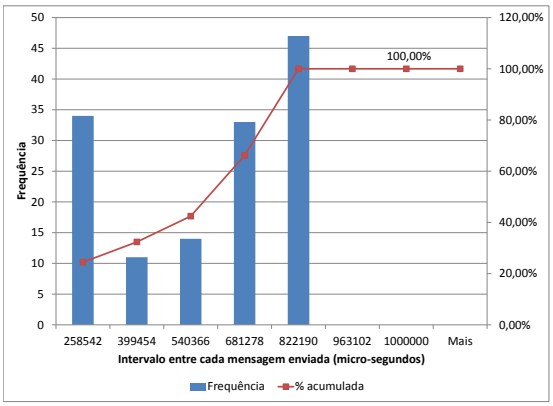
Aqui são apresentados os resultados relativos *timings* de envio e receção de mensagens, bem como o tempo de construção de cada mensagem enviada e o tempo de processamento de cada mensagem recebida.

Em primeiro lugar apresentam-se os resultados em relação ao envio de mensagens CAM em cada cenário de teste. Como se pode verificar através dos gráficos da figura 6.5 e da tabela 6.1 que o intervalo máximo de envio de mensagens CAM aumenta com a carga de processamento no *Raspberry Pi* aumenta: Nos cenários 2 e 3, o intervalo máximo situa-se nos 800ms, mas no primeiro cenário 1 esse valor atinge os 1,3s, ficando, portanto fora do

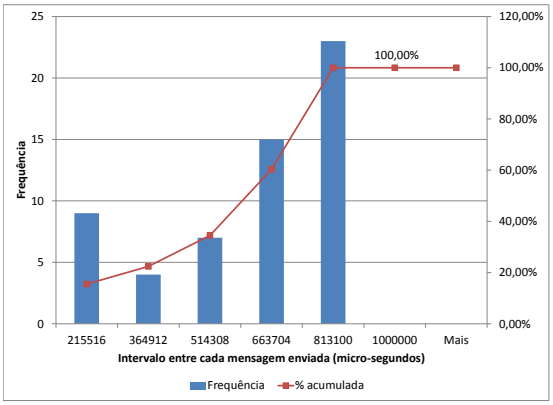
definido pelo standard do ETSI. A construção destas mensagens juntamente com mensagens DENM a uma frequência de 33Hz provoca uma elevada carga no Raspberry Pi em termos de alocação de memória e de tratamento de informação, além disso o esforço que a encriptação de cada mensagem (módulo de segurança) exige ao SBC leva a estes resultados. Na figura 6.6 verificou-se que o módulo de encriptação (comando "`./1609dot2`"), no caso de um grande volume de mensagens, é o processo que ocupa mais tempo de CPU. Na mesma figura também se pode ver que a combinação dos três módulos a correr no SBC ao mesmo tempo leva a que o CPU funcione, na maior parte do tempo, a 100% de capacidade, isto pode indicar que existe trabalho a fazer em termos de otimização do *software* ou então que o SBC não tem o desempenho suficiente para este tipo de aplicações.



(a) Cenário de Teste 1 – Carga elevada na Rede



(b) Cenário de Teste 2 – Carga Moderada na Rede



(c) Cenário de Teste 3 – Carga Ligeira na Rede

Figura 6.5: *Timings* de envio de cada mensagem CAM para os vários cenários

Tabela 6.1: Comparação de resultados médios de cada cenário.

Cenário	Intervalo Médio (μs)	Intervalo Máximo (μs)
1	573660	1306130
2	508850	822190
3	537876	813100

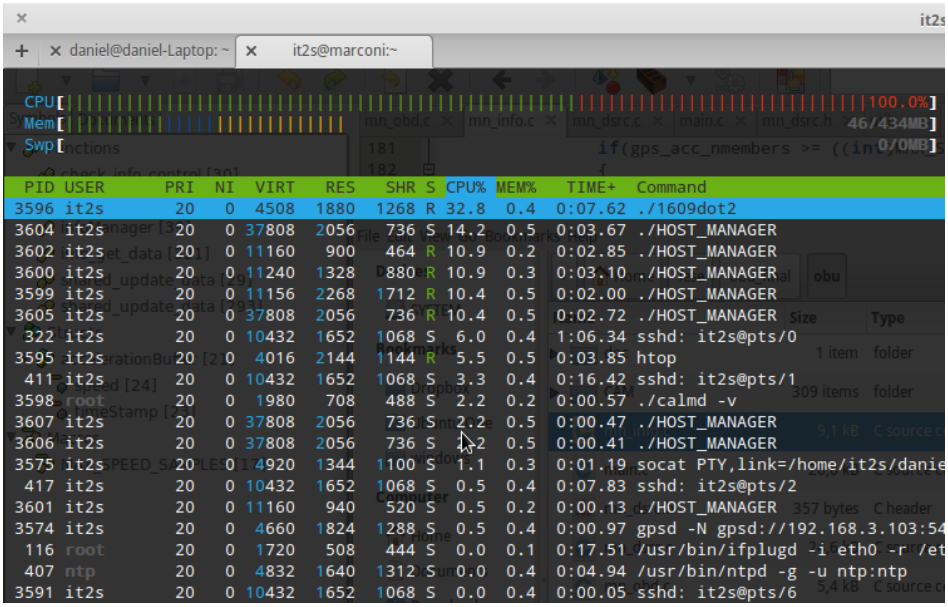


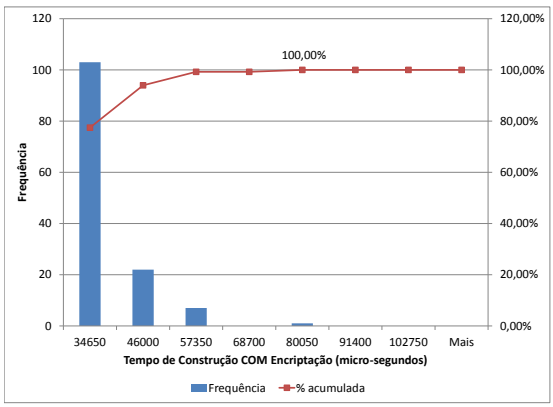
Figura 6.6: Processos em execução no decorrer no Cenário 1, por ordem de tempo de CPU

Ainda assim, olhando para os dois últimos cenários – cenários mais comuns em ambiente real – verifica-se que os sistema cumpre os standards ETSI em termos de intervalo entre o envio de mensagens CAM.

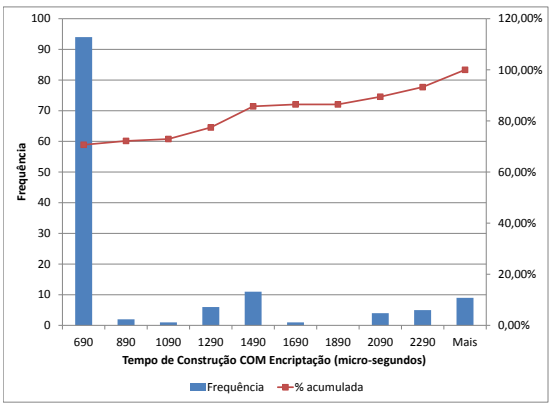
Em seguida, pegando nos dados obtidos para o cenário 2, mediu-se o tempo que cada mensagem CAM leva a ser construída, isto é desde quando é acionado o *trigger* para construção da mensagem até que esta é enviada para camada de rede e também o tempo que uma mensagem recebida leva a ser processa e os seus dados expostos ao sistema – gráficos da figura 6.7 e tabela 6.2.

Tabela 6.2: Tempos de processamento de cada CAM, cenário 2.

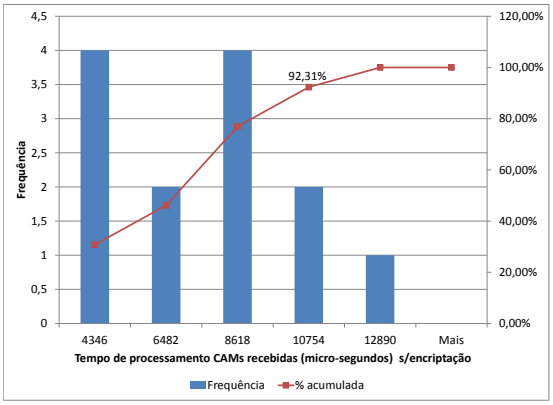
CAMs	Tempo Médio (μs)	Tempo Máximo (μs)
Construção (c/encript)	31561	80050
Construção (s/encript)	1310	32140
Receção (s/encript)	6841	12890



(a) Tempo de construção de CAMs a enviar, com encriptação.



(b) Tempo de construção de CAMs a enviar, sem encriptação.



(c) Tempo de processamento de CAMs recebidas, sem encriptação

Figura 6.7: Tempos de construção de mensagens CAM enviadas e processamento de recebidas.

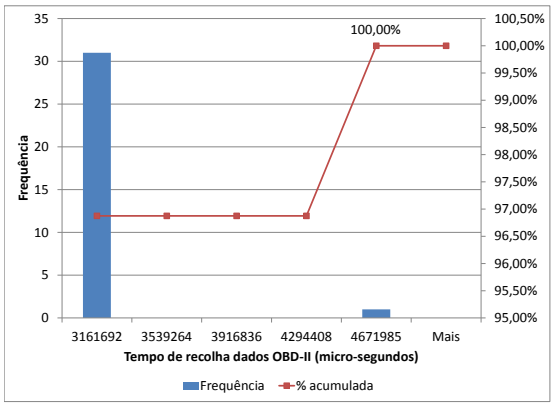
Em termos de construção das mensagens, verifica-se, novamente, que o módulo de segurança introduz um atraso significativo no sistema. Incluindo o tempo gasto na encriptação na mensagem obteve-se um tempo de construção médio de, aproximadamente, 32ms, deduzindo esse tempo (figura 6.7(b)) o tempo de construção médio reduz-se para cerca de 1,4ms. Quer num caso quer no outro obtiveram-se tempos dentro do que o standard ETSI, isto é, 50ms (figura 3.2 do ponto *Requisitos Temporais e Regras para Construção e Envio de Mensagens CAM*).

O tempo processamento de mensagens recebidas é bastante menor, bastando, em média, apenas 6,8ms para que a mensagem seja descodificada e exposta ao sistema, isto descontando o tempo de desencriptação.

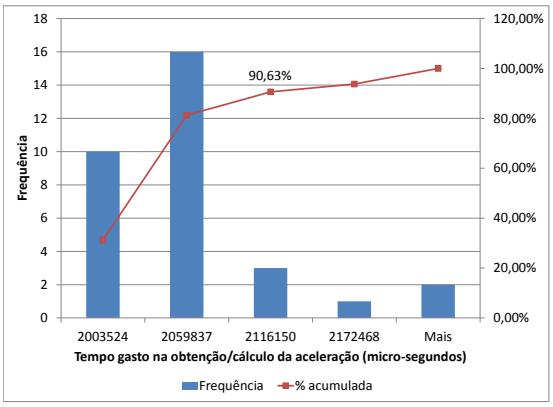
6.7.2 Aquisição de Dados

Neste ponto apresentam-se os resultados relativos ao tempo decorrido na aquisição de dados dos sensores, mais especificamente, do recetor GPS e do OBD-II, utilizando os valores recolhidos no cenário 2.

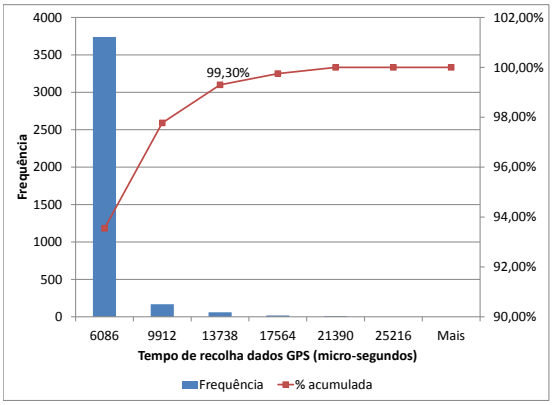
Na figura 6.8 e na tabela 6.3 são então apresentados os tempos de recolha dos dados de posicionamento (GPS) e dos sensores do veículo (OBD-II).



(a) Tempo de recolha dados OBD.



(b) Tempo gasto na recolha/cálculo da aceleração (OBD)



(c) Tempo de recolha dados GPS.

Figura 6.8: Tempo despendido na recolha de dados.

Através dos gráficos da figura 6.8 e da tabela 6.3 verifica-se que o período de atualização de dados GPS é extremamente baixo (≈2,4ms), revelando que os dados GPS estão sempre muito atuais. Pelo contrário, os dados OBD-II revelam um atraso de aproximadamente 3 segundos, em que 2 desses segundos são gastos a calcular a aceleração. Isto revela uma forte

Tabela 6.3: recolha de dados por sensor.

	Tempo Médio (μs)
Dados GPS	2370
OBD-II	2901000
OBD-II (só aceleração)	2033678

atraso na recolha dos dados OBD-II. A aceleração, como é descrito em 5.4.1, é calculada a partir de dez valores de velocidade, então pode-se estimar que o tempo de resposta do simulador ODBsim a cada pedido é de, aproximadamente, 0,2 segundos, extremamente lento. Antes deste foi utilizado o simulador Scantool.net ECUsim 2000, verificando-se que este tinha tempos de resposta na ordem das unidades de mili-segundo, um teste utilizando esse simulador juntamente com o Scantool.net ElmScan 5 seria mais próximo da realidade dos interfaces OBD-II dos veículos atuais, com tempos de reposta muito mais reduzidos.

Em suma, este atraso na recolha dos dados OBD-II pode-se justificar pela fraca resposta do simulador OBDsim, num automóvel comum isto não se verificará. Quanto ao GPS verifica-se que a recolha de dados ocorre a uma frequência elevada, permitindo que o sistema disponha sempre dos dados mais atuais.

6.8 Comunicação USB - Smartphone Android e Monitorização da Plataforma

Em relação à comunicação USB com o smartphone Android e à funcionalidade de monitorização da plataforma, fizeram-se testes simples de cariz empírico para assegurar o seu funcionamento básico.

Em primeiro lugar, a implementação da comunicação USB com o smartphone permite o envio de dados OBD-II da OBU para o smartphone. Através de um teste simples verificou-se que, durante a transmissão, a informação relativa a sensores do veículo mostrada no ecrã do *smartphone* era, efetivamente, a mesma que a OBU apresentava, variando também da mesma maneira. Isto indicou que a implementação foi bem sucedida. Também bem sucedida foi a implementação da troca de mensagens de alerta entre o *smartphone* e a OBU. Uma mensagem de alerta gerada no *smartphone* é imediatamente mostrada no ecrã da OBU, bem como os restantes dados relativos a esse alerta: posição e tipo de acidente.

Ainda em relação à comunicação Smartphone - OBU, acresce dizer que toda esta troca de informação funcionou de forma *plug-and-play*, isto é, o utilizador pode ligar e desligar o

interface USB como entender que a comunicação automaticamente reestabelecida sem que haja a necessidade de reiniciar a aplicação do *smartphone* ou a aplicação da OBU.

Por fim, verificou-se o correto funcionamento da monitorização da plataforma utilizando uma aplicação externa ao sistema. Esta imprime periodicamente o conteúdo da base de dados para onde as estações reportam o seu estado e o estado das estações vizinhas. Verificou-se então que existe coerência entre os dados armazenados nessa base de dados e o que cada estação estava a imprimir no ecrã.

Capítulo 7

Conclusão e Trabalho Futuro

7.1 Conclusões

O foco principal deste trabalho relacionou-se com as mensagens produzidas e tratadas na camada de aplicação e suporte à aplicação ITS. Implementou-se então *software* capaz de recolher informação de sensores (inclusive sensores do próprio veículo), construir, enviar e receber mensagens na rede veicular. Além disso, introduziu-se no sistema a capacidade de comunicação com um *smartphone* para interação com o utilizador e ainda um sistema básico de monitorização das estações em funcionamento.

Depois de analisada a implementação efetuada e os resultados obtidos conclui-se que, apesar de ser um primeiro protótipo, o sistema implementado consegue cumprir, na sua maioria, com o formato e os *timings* básicos definidos pelo ETSI em relação a mensagens CAM e DENM e, mesmo contendo sensores lentos (OBD-II – OBDsim), o sistema foi capaz de construir e envias as mensagens nos *timings* corretos, o que revela que a arquitetura do *software* é adequada.

Neste tipo de sistemas computacionais – SBCs – a otimização do sistema deverá ser uma prioridade, tarefas como alocação e de-alocação de memória frequentemente e processos constantemente em estado de "*busy-waiting*"podem prejudicar o gravemente o desempenho do sistema. Ainda sobre o tema do desempenho da plataforma e relacionando com a integração que foi efetuada, conclui-se que um módulo de encriptação não deverá ser executado no processador do SBC Raspberry Pi uma vez que consome bastantes recursos e degrada o desempenho do sistema devido às tarefas "pesadas"que desempenha. A solução para este problema passaria pelo uso de um co-processador dedicado à encriptação da informação ou a utilização de um SBC com processador mais poderoso.

Claro que se trata de um sistema/protótipo inicial e que seria impossível implementar, exatamente, todos os requisitos e funcionalidades que os standards exigem, mas, no geral, o sistema implementado cumpre os objetivos a que propõe relativamente à gestão de mensagens, comunicação com o *smartphone* e monitorização.

7.2 Trabalho Futuro

Partindo do trabalho efetuado nesta dissertação e das conclusões a que se chegou, verifica-se que, em termos de trabalho futuro, existem muitos temas a explorar:

- Implementação de um sistema de atualização automática da plataforma baseada na rede de backup 3G/4G. Como se trata de *software* existem sempre correções e adição de funcionalidades que não deveriam necessitar que o equipamento fosse desligado e/ou que o utilizador se deslocasse a um local especializado. Estas atualizações poderiam ser distribuídas e instaladas de forma remota, automática e transparente ao utilizador.

- Nesta implementação existem alguns campos das mensagens CAM e DENM que não puderam ser preenchidos por falta de sensores no sistema. Assim, a adição de mais sensores ao sistema e/ou exploração mais profunda dos sensores presentes nos automóveis atuais deverá ser realizada de forma que as mensagens tenham informações mais completas.

- Analisar o código fonte do *software* desenvolvido nesta dissertação e tentar perceber se este precisa de ser otimizado ou se, realmente, o SBC Raspberry Pi não tem capacidade de processamento suficiente para este tipo de tarefas.

Apêndice A

Estrutura Mensagem CAM

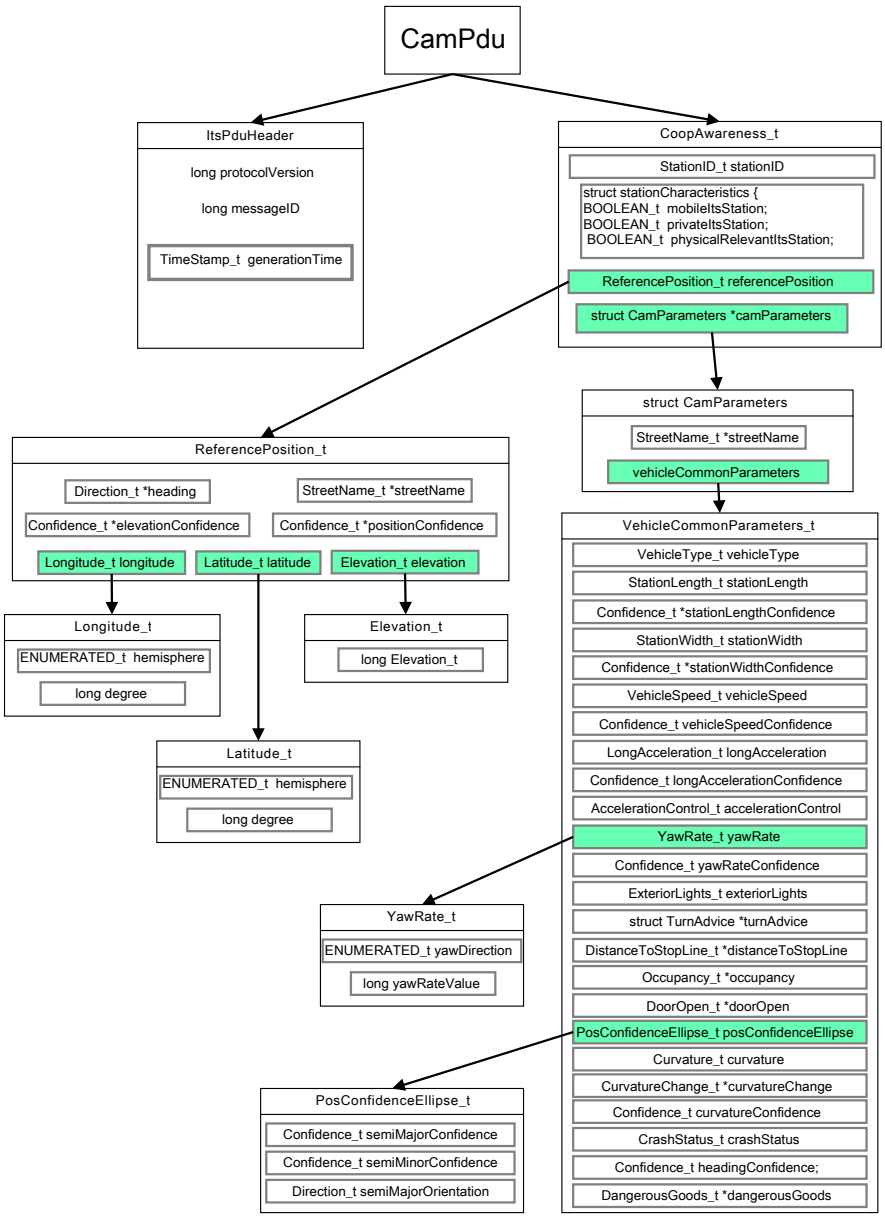


Figura A.1: Relação campos de uma mensagem *Cooperative Awareness Message* (CAM) em C (campos com cujo o nome começa por um asterisco não são obrigatórios)

Bibliografia

[1] M. Peden et al. World report on road traffic injury prevention, 2004.

[2] M.D. Meyer, C. Systematics, and American Automobile Association. *Crashes Vs. Congestion: What's the Cost to Society?.*, chapter ES. American Automobile Association, 2008.

[3] Hassnaa Moustafa and Yan Zhang. *Vehicular Networks: Techniques, Standards, and Applications.* Auerbach Publications, Boston, MA, USA, 1st edition, 2009. ISBN 1420085719, 9781420085716.

[4] P. Papadimitratos, A. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza. Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation. *Communications Magazine, IEEE*, 47(11):84 –95, november 2009. ISSN 0163-6804. doi: 10.1109/MCOM.2009.5307471.

[5] H. Hartenstein, K. Laberteaux, and Inc Ebrary. *VANET: vehicular applications and inter-networking technologies.* Wiley Online Library, 2010.

[6] John B Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.

[7] R. Baldessari, B. Bödekker, M. Deegener, A. Festag, W. Franz, C.C. Kellum, T. Kosch, A. Kovacs, M. Lenardi, C. Menig, et al. Car-2-car communication consortium-manifesto. *DLR Electronic Library [http://elib. dlr. de/perl/oai2](Germany)*, 2007.

[8] ES ETSI. 202 663. *European profile standard for the physical and medium access control layer of intelligent transport systems operating in the*, 5, 2010.

[9] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil.

Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. *Communications Surveys & Tutorials, IEEE*, (99):1–33, 2011.

[10] *Decisão da comissão de 5 de Agosto de 2008 relativa à utilização harmonizada do espectro radioeléctrico na faixa de frequências de 5875-5905 MHz para aplicações relacionadas com a segurança no domínio dos sistemas de transporte inteligentes (STI)*, Bruxelas, Agosto 2008. Comissão Europeia.

[11] FCC allocates spectrum 5.9 GHz range for intelligent transportation systems uses. URL http://transition.fcc.gov/Bureaus/Engineering_Technology/News_Releases/1999/nret9006.html. Agosto 2012.

[12] Kevin C Lee, Uichin Lee, and Mario Gerla. Survey of routing protocols in vehicular ad hoc networks. *Advances in Vehicular Ad-Hoc Networks: Developments and Challenges, IGI Global*, 21, 2009.

[13] Car to Car Communication Consortium et al. C2c-cc manifesto, version 1.1, 2007.

[14] *IoT CoAP PlugtestsTM and Workshop*, 2012. Intelligent Cooperative Sensing for Improved traffic efficiency, Intelligent Cooperative Sensing for Improved traffic efficiency.

[15] M. Ferreira, R. Fernandes, H. Conceição, W. Viriyasitavat, and O.K. Tonguz. Self-organized traffic control. In *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*, pages 85–90. ACM, 2010.

[16] R. Uzcategui and G. Acosta-Marum. Wave: a tutorial. *Communications Magazine, IEEE*, 47(5):126–133, 2009.

[17] D. Jiang and L. Delgrossi. Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 2036–2040. Ieee, 2008.

[18] EN ETSI. 302 665: Intelligent transport systems (its). *Communications Architecture*, 2010.

[19] O. Brickley, C. Shen, M. Klepal, A. Tabatabaei, and D. Pesch. A data dissemination strategy for cooperative vehicular systems. In *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, pages 2501–2505. IEEE, 2007.

[20] Work programme - work item schedule detailed report. URL http://webapp.etsi.org/workprogram/Report_Schedule.asp?WKI_ID=31022.

[21] TR ETSI. 102 638 v1. 1.1.(2009). intelligent transport systems (its); vehicular communications; basic set of applications; definitions. Technical report, ETSI technical report, June, .

[22] F. Kargl, P. Papadimitratos, L. Buttyan, M. Muter, E. Schoch, B. Wiedersheim, T.V. Thong, G. Calandriello, A. Held, A. Kung, et al. Secure vehicular communication systems: implementation, performance, and research challenges. *Communications Magazine, IEEE*, 46(11):110–118, 2008.

[23] @ua_online > jornal. URL http://uaonline.ua.pt/detail.asp?c=24498. Agosto 2012.

[24] TeK > notícias > computadores > táxis do porto ligados em rede. URL http://tek.sapo.pt/noticias/computadores/taxis_do_porto_ligados_em_rede_1256363.html. Agosto 2012.

[25] Investigadores do porto querem colocar semáforos dentro dos carros (vídeo) - exame informática. URL http://exameinformatica.sapo.pt/noticias/ciencia/2010/08/26/investigadores-do-porto-querem-colocar-semaforos-dentro-dos-carros-video. Agosto 2012.

[26] Car2car members. URL http://www.car-to-car.org/index.php?id=members&L=llkptclqq.

[27] ICSI - intelligent cooperative sensing for improved traffic efficiency. URL http://www.ict-icsi.eu/description.html#.UhEEcazCiXo.

[28] Brisa innovation. URL http://www.brisainovacao.pt/en/innovation/projects/headway. Agosto 2012.

[29] ITUT Rec. 691 (2002)— iso/iec 8825-2: 2002 “information technology–asn. 1 encoding rules: Specification of packed encoding rules (per)”.

[30] TS ETSI. 102 637-2 v1. 2.1. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*, 2011.

[31] TS ETSI. 102 637-3 v1. 1.1. *Technical Specification Intelligent Transport Systems (ITS)*, .

[32] Arnaldo S. R.Oliveira. It2s board description. Instituto de Telecomunicações - Pólo de Aveiro, March 2013.

[33] Gumstix® overo® IronSTORM COM. URL https://www.gumstix.com/store/product_info.php?products_id=268. Setembro 2012.

[34] Broadcom Corporation. *BCM2835 ARM Peripherals*, 2012.

[35] Raspberry pi review & rating | PCMag.com, . URL http://www.pcmag.com/article2/0,2817,2407058,00.asp. Agosto 2012.

[36] RPi low-level peripherals - eLinux.org. URL http://elinux.org/Rpi_Low-level_peripherals. Setembro 2012.

[37] Rpi hardware - elinux.org. URL http://elinux.org/RPi_Hardware. Agosto 2013.

[38] Raspberry pi impressions: the $35 linux computer and tinker toy – engadget, . URL http://www.engadget.com/2012/06/01/raspberry-pi-impressions-the-35-linux-computer-and-tinker-toy/. Setembro 2012.

[39] FAQs | raspberry pi. URL http://www.raspberrypi.org/faqs. Agosto 2012.

[40] RPi VerifiedPeripherals - eLinux.org. URL http://elinux.org/RPi_VerifiedPeripherals.

[41] GPSd — put your GPS on the net!, . URL http://catb.org/gpsd/index.html.

[42] Elm electronics products - OBD ICs. URL http://www.elmelectronics.com/obdic.html.

[43] Elm Electronics. *ELM327 Datasheet*. Elm Electronics.

[44] ScanTool.net LLC - ElmScan 5 USB - ScanTool.net, . URL http://www.scantool.net/elmscan-5-usb.html.

[45] ScanTool.net, LLC - downloads, . URL http://www.scantool.net/scantool/downloads/archive/diagnostic-software/.

[46] Open source ASN.1 compiler: asn1c 0.9.24. URL http://lionet.info/asn1c/compiler.html.

[47] ScanTool.net LLC - OBD simulator: ECUsim 2000 multiprotocol OBD-II ECU simulator - ScanTool.net, . URL http://www.scantool.net/ecusim-2000.html.

[48] OBDSim, . URL http://icculus.org/obdgpslogger/obdsim.html.

[49] OBDsim man page, . URL http://icculus.org/obdgpslogger/manpages/render/obdsim.txt.

[50] gpsfake, . URL http://catb.org/gpsd/gpsfake.html.

[51] socat. URL http://www.dest-unreach.org/socat/.